Learning condition–action rules for personalised journey recommendations*

Matthew R. Karlsen and Sotiris Moschoyiannis

Department of Computer Science, University of Surrey, GU2 7XH
matthew.r.karlsen@surrey.ac.uk, s.moschoyiannis@surrey.ac.uk

Abstract. We apply a learning classifier system, XCSI, to the task of providing personalised suggestions for passenger onward journeys. Learning classifier systems combine evolutionary computation with rule-based machine learning, altering a population of rules to achieve a goal through interaction with the environment. Here XCSI interacts with a simulated environment of passengers travelling around the London Underground network, subject to disruption. We show that XCSI successfully learns individual passenger preferences and can be used to suggest personalised adjustments to the onward journey in the event of disruption.

Keywords: rule-based machine learning · XCSI · passenger preferences

1 Introduction

Modern route recommendation systems suggest multiple routes, times and modes, with near instant results. However, unique passenger preferences are ignored. Here we use the Learning Classifier System [14] XCSI [17] (explained in Section 2) to learn individual transport mode preferences, given the current state of the transport network and other factors (e.g. weather). The idea is to provide advice as part of a "recommendation engine" that pro-actively suggests adjustments to a journey as data becomes available.

The remainder of this paper is structured as follows. XCSI and its application to making *personalised* recommendations is described in Section 2. Experiments, described in Section 3, are performed in relation to the above challenge. The results and accompanying discussion for these experiments is presented in Section 4. Related work is outlined in Section 5. The paper concludes in Section 6.

2 Applying XCSI to provision of recommendations

The overall challenge is to provide a single integer recommendation (representing a mode or modes) for each unique situation (combination of passenger preferences, current context and environment state). The XCSI system used to achieve

* This research was partly funded by the Department for Transport, via Innovate UK and the *Accelerating Innovation in Rail* (AIR) Round 4 programme, under the *Onward Journey Planning Assistant (OJPA)* project.

this is built in two steps. We first implement XCS [15,16] according to the detailed algorithmic specification supplied by Butz and Wilson [2]. Following from this, the XCSI modification is implemented based on [17]. We use XCSI (with integer, rather than boolean variables) over XCS here because a number of factors relevant to onward journey recommendations have more than two possible states. Due to space limitations we refer to [8] for the full XCSI specification.

A learning classifier system possesses sensors, internal mechanisms of operation, and effectors. The system also possesses a 'feedback' mechanism to judge the impact of effector-implemented actions on the environment relative to the system's goals. The system adapts its internal structure over time to effectively model the environment with the aim of achieving these goals. The central system operation is the following: **(1)** detect the 'environment state' via the detectors, **(2)** convert this state in to an integer string (e.g. 42403015), **(3)** determine an integer-labelled action based on LCS internal structure, **(4)** implement the action in the environment, **(5)** obtain feedback via the feedback sensor, **(6)** use feedback to adapt internal structure to model the environment, **(7)** re-start at (1). In order to operate in the above manner XCSI is put together from a number of components, governed by a central algorithm (see [8] for more detail).

2.1 Detectors

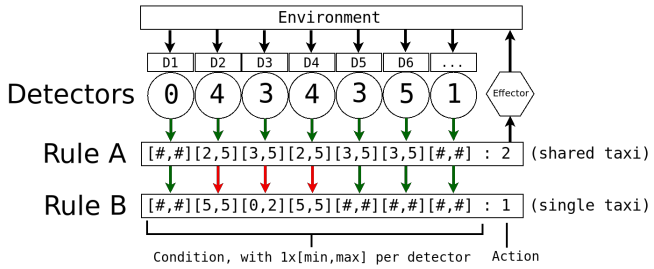
The *situation* is comprised of environment factors, journey-specific factors the needs and preferences of the passenger. Here, our situation is comprised of a number of **environment factors** (Train QoS (quality of service) [0 or 5], Taxi QoS [0 or 5], Tube QoS [0 or 5], Boat QoS [0 or 5], Bus QoS [0 or 5]), **journey-specific factors** (delay on current route [0 or 5], delay on current mode [0 or 5], onward delay [0 or 5]), 4 relatively-invariant **passenger preferences** (value preference [0 to 5], speed preference [0 to 5], comfort preference [0 to 5], shelter preference [0 to 5]), and weather [0 to 5] (cf see Fig. 1)

By treating passenger preferences as external we treat the system as a kind of 'oracle'. An alternative would be to have one system per passenger and omit the passenger preference detectors. This would enable finer-tuned recommendations after many steps but would also mean slower learning (experience would not be shared between passengers) and greater resource costs. With the 'oracle' approach, the feedback acquired through use can be shared between passengers and *passengers still differ* due to their different ratings for each preference.

2.2 Population of Rules

The central component of the XCSI system is a population of IF <condition> THEN <action> rules of size N. Given the above inputs, we can construct a series of rules to relate given situations to particular transport 'actions' (travel options). For the current problem, XCSI takes as *input* a list of integers, where each integer corresponds to one of the above properties. The *output* at each step is a single integer representing the action that should be taken in the environment (or, in the present case the *recommended* travel mode), as shown in Fig. 1. The

Fig. 1. The structure of the environment detectors and condition–action rules.



condition side of the rule represents the environment state that triggers the rule. The '#' symbols represent 'don't care', indicating that any detector integer value in this position is acceptable as part of a rule match. The right hand side of the rule is an action, represented as an integer.

2.3 Rule Matching

For a given integer string input some of the rules match whilst others do not, as shown in Figure 1, where we have two rules, seven detectors, and an effector. Each detector takes a value in the range $[0, 5]$. Here the inputs read in are 0,4,3,4,3,5,1. Each of these detector values is compared with a related range within each rule. The [min,max] range indicates whether the particular rule matches the given input at that particular detector. All ranges must match the relevant detectors for the rule to match fully. As we can see in the example, rule 'A' matches but rule 'B' does not. Rule A corresponds to action 2 (recommend a shared taxi) and thus in this situation the recommended action supplied to the user (in the environment) is to book a shared taxi.[1]

2.4 Effector and Feedback Mechanism

In a real-world system the effector would display the suggested onward travel option via a passenger interface. In the experiments described herein, an integer is supplied to the simulation environment which determines whether the recommendation was correct or not. Possible onward journey modes that can be suggested to the passenger for short to medium distance journeys are numerous. Virtual 'journeys' are also possible [18]. Here we limit ourselves to no change (**0**), single taxi (**1**), shared taxi (**2**), bus (**3**), boat or water bus (**4**), underground or tube (**5**), and regular train (**6**). For the simulation herein, the feedback mechanism provides a payoff of **1000** for a correct answer (i.e. the simulated passenger's preference is indeed for the suggested action) and **0** for an incorrect answer.

[1] Note that in the simple example here we depict a match as immediately triggering the rule in question – the actual action selection is more complex.

3 Experiments

The simulation involves 300 random passengers, *with individual preferences*, origin locations and destination locations. A number of planned journeys on the London Underground are generated, using passenger origins and destinations, exceeding the number of steps XCSI will run for. The shortest path for each journey is calculated using the A* implementation from GraphStream [5].

For each time step, 5% of links are randomly selected and marked as 'out-of-order'. If any of the links on the shortest path between the current node and the journey destination are out-of-order then the delay property is set to 5. If none are out of order it is set to 0. The value of the tube QoS detector is set to 0 if any links on the shortest path are out-of-order; otherwise it is set to 5.

The abstract 'start time' of each journey is randomly generated in the range $[0, 99]$. For each time step the weather property (0 to 5) is randomly generated. The availability of train, boat, bus and taxi is exogenous to the simulation and simply marked as either available (5) or unavailable (0). Once the journeys generation is complete their order is shuffled. The first journey with the minimum start time is then used to create the first state, the next journey with the same start time is used to create the second state, and so on. This process is then repeated with all the entries for all time steps, until all the states have been added to the final state sequence. It is this sequence that is used each time XCSI requests a state. The 'preference table' that relates consumer preferences to particular preferred actions is input as static. A sub-set is shown in Table 1.

Table 1. A sample of condition-action rules. From left to right, the conditions are the input factors described in Sec. 2.1. The suggested actions were described in Sec. 2.4.

Condition	: Action
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][#,#][2,5][#,#][#,#] :	1
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][2,5][#,#][#,#][#,#] :	1
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][#,#][4,5][#,#][#,#] :	1
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][4,5][#,#][#,#][#,#] :	1
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][0,1][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][2,3][0,1][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][2,3][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][0,1][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][2,3][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][0,1][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][0,1][4,5][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][2,3][2,3][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][2,3][4,5][#,#][#,#] :	2
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][0,1][0,1][0,1][#,#][#,#] :	3
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][0,1][0,1][#,#][#,#] :	3
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][2,3][2,3][0,1][#,#][#,#] :	3
[0,0][0,0][0,0][5,5][5,5][#,#][#,#][5,5][4,5][0,1][0,1][#,#][#,#] :	3

3.1 Parameters

The original parameter settings from [2] are shown in the Default column while variations of these are shown in the Other Values column in Table 2.

Table 2. The parameter settings used.

Parameter	Default	Other Values
N	15600	11700, 19500
θ_{del}	20	10, 30
δ	0.1	0.05, 0.2
γ	0.71	N/A
θ_{mna}	7	N/A
$P_{\#}$	0.55	0.35, 0.75
r_0	2	1, 3
p_I	10	N/A
ϵ_I	0	N/A
ϵ_0	0	5, 20
F_I	10	N/A
ϵ_0	10	N/A
θ_{sub}	20.0	10, 60
doActionSetSubsumption	true	false
doGaSubsumption	true	false
θ_{ga}	50	25, 150
χ	0.8	0.6, 1.0
μ	0.04	0.02, 0.08
m_0	1	2, 3
p_{explr}	0.5	0.25, 0.75
α	0.1	0.08, 0.12
ν	5.0	4.0, 6.0
β	0.15	0.1, 0.2
activateExploreAndExploitMode	true	false

3.2 Simulation Runs

For each combination of parameters in the table above (varying a single parameter away from default each time) we perform one 'experiment'. For each experiment we perform 32 repetitions, recording the minimum, maximum and average number of incorrect predictions across these repetitions.

For each experiment repetition we first set up the simulation environment. XSCI is then initialised with the required parameters. XCSI is then run for a number of steps as specified by the parameters (typically 50,000 steps). The evolved rules are then output. Next, p_{explr} is set to 0.0 and the feedback mechanism, action set updater and genetic algorithm are disabled (essentially all adaptation is disabled). XCSI then runs for a further 1000 steps and records the number of errors (incorrect suggestions) over these steps. The input from the training phase is *not* re-used in the test phase.

4 Results and Discussion

The results for the experiments are shown in Table 3. The average error level for

Table 3. The results for the experiments, with 32 trials per experiment.

Parameter	Value	Min. Error %	Avg. Error %	Max. Error %
Defaults	see above	0.008	0.031	0.071
trials	25000	0.031	0.061	0.1
trials	75000	0.005	0.020	0.066
N	11700	0.006	**0.028**	0.082
N	19500	0.013	0.030	0.059
activateExploreAndExploit	false	0.051	0.089	0.236
θ_{del}	10	0.012	**0.027**	0.052
θ_{del}	30	0.009	0.029	0.079
δ	0.05	0.009	**0.027**	0.06
δ	0.2	0.009	0.031	0.072
$P_{\#}$	0.35	0.004	**0.019**	0.042
$P_{\#}$	0.75	0.025	0.063	0.137
r_0	1	0.008	**0.027**	0.055
r_0	3	0.015	0.029	0.045
ϵ_0	5	0.008	**0.024**	0.051
ϵ_0	20	0.018	0.044	0.129
θ_{sub}	10	0.007	0.038	0.077
θ_{sub}	60	0.01	**0.027**	0.06
doActionSetSubsumption	false	0.0	**0.010**	0.02
doGaSubsumption	false	0.011	0.033	0.065
θ_{ga}	25	0.006	**0.019**	0.05
θ_{ga}	150	0.022	0.071	0.156
χ	0.6	0.011	0.034	0.064
χ	1	0.011	**0.027**	0.055
μ	0.02	0.018	0.055	0.113
μ	0.08	0.004	**0.021**	0.072
m_0	2	0.005	0.036	0.088
m_0	3	0.013	0.038	0.081
p_{explr}	0.25	0.009	0.038	0.081
p_{explr}	0.75	0.018	0.064	0.139
α	0.08	0.004	**0.027**	0.056
α	0.12	0.008	0.028	0.071
ν	4	0.008	0.030	0.076
ν	6	0.009	**0.026**	0.071
β	0.1	0.004	**0.018**	0.039
β	0.2	0.023	0.054	0.117

the 1000 test steps for the parameter settings used (with 50,000 'training' steps) is 3.1%. The number of errors falls as the number of training steps increases, from 6.1% errors with 25,000 steps to 2.0% per 75,000 steps. This is to be expected since logically the greater the number of learning iterations, the better the understanding of the passenger preferences (if the system is working as expected). Upon completion of these results an additional run was performed, combining the parameter setting shown in bold in Table 3. The results are minimum error 0.001%, average error 0.00734%, and maximum error 0.02%.

5 Related Work

Present-day 'live' services such as Citymapper (citymapper.com) suggest routes, times and modes, but do not consider passenger preferences. Research in this direction is limited, with recent work proposing Bayesian networks [4], heuristics and traditional routing [1] and 'case-based reasoning' [9,6], which provide customised recommendations without actual specification of passenger preferences.

In previous work [8] we have applied the XCS variant [2] of LCS to study *controllability* [10] in Random Boolean Networks. The extension with integer-based conditions in XCS*I* is necessary to capture passenger preferences faithfully.

Specifically, XCSI is suited to the current problem for the following reasons: (1) the rules produced are human-readable making the gathered knowledge available for analysis, (2) they are on-line, hence can provide rapid responses manner (providing single input–output iterations) in contrast to batch-based approaches that require a number of training instances, (3) the system adapts to changes within the mapping of input states to preferred actions (i.e. the system can cope with concept drift), (4) the system is able to construct thousands of rules without direct human input which tend to be time consuming and prone to error.

Pulugurta et al. [12] consider classifiers for predicting travel mode and find that the fuzzy logic model has superior performance over the multinomial logit model. Omrani [11] find that neural network-based approaches (multi layer perceptron and radial basis function networks) have higher performance than multinomial logistic regression and support vector machines. Sekhar et al. [13] find that the random forest classifier out-performs the multinomial logit model. Hagenauer et al. [7] compare seven classifiers applied to the task of predicting travel mode based on a number of inputs. They find that the Random Forest classifier produces the best performance of the seven. To the best of our knowledge XCSI has not been applied before to the provision of onward journey recommendations.

6 Concluding remarks and future work

XCSI represents a comparatively novel approach to constructing an onward journey recommendation system. Our results in Section 4 indicate that an error rate of 3.1% is achievable with the default parameter settings. With adjusted parameter settings we find that the error rate is reduced yet further to just 0.734% on average. In this way, XCSI is demonstrably able to develop a relatively compact set of rules used to provide accurate recommendations to simulated travellers.

Directions for future work include more precise conditions for detectors with binary states, code optimisation, an XCSI extension to be used in a parallel by multiple passengers, an evaluation of the use of supervised learning rather than reinforcement learning, and implementation of multi-modal solutions.

Additionally we note that DMN (Decision Models and Notation)-based rules [3] may become prohibitively complex for a human to construct (with a large number of conditions or rules). It would be possible, particularly in conjunction with the messy encoding mentioned above, to evolve DMN rules using XCS.

This approach could produce complex and adaptive rule sets without the need for human intervention in the system. Alternatively, the approach used to merge rules in [3] could well be used as a mechanism for rule set reduction in XCSI.

References

1. Bucher, D., Jonietz, D., Raubal, M.: A Heuristic for Multi-modal Route Planning. In: Progress in Location-Based Services 2016, pp. 211–229. Springer (2017)
2. Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. In: International Workshop on Learning Classifier Systems. pp. 253–272. Springer (2000)
3. Calvanese, D., Dumas, M., et al: Semantics, analysis and simplification of DMN decision tables. Information Systems (2018), in press
4. Campigotto, P., Rudloff, C., Leodolter, M., Bauer, D.: Personalized and situation-aware multimodal route recommendations: the FAVOUR algorithm. IEEE Transactions on Intelligent Transportation Systems **18**(1), 92–102 (Jan 2017)
5. Dutot, A., Guinand, F., et al: Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In: Emergent Properties in Natural and Artificial Complex Systems (2007)
6. Ginty, L.M., Smyth, B.: Collaborative Case-Based Reasoning: Applications in Personalised Route Planning. In: Case-Based Reasoning Research and Development, vol. 2080, pp. 362–376. Springer (2001)
7. Hagenauer, J., Helbich, M.: A comparative study of machine learning classifiers for modeling travel mode choice. Expert Systems with Applications **78**, 273–282 (2017)
8. Karlsen, M.R., Moschoyiannis, S.: Evolution of control with learning classifier systems. Journal of Applied Network Science (2018), in press
9. McGinty, L., Smyth, B.: Personalised route planning: A case-based approach. In: European Workshop on Advances in Case-Based Reasoning. pp. 431–443. Springer, Berlin, Heidelberg (2000)
10. Moschoyiannis, S., Elia, N., Penn, A., et al: A web-based tool for identifying strategic intervention points in complex systems. In: Proc. Games for the Synthesis of Complex Systems. EPTCS, vol. 220, pp. 39–52 (2016)
11. Omrani, H.: Predicting travel mode of individuals by machine learning. Transportation Research Procedia **10**, 840–849 (2015)
12. Pulugurta, S., Arun, A., Errampalli, M.: Use of artificial intelligence for mode choice analysis and comparison with traditional multinomial logit model. Procedia-Social and Behavioral Sciences **104**, 583–592 (2013)
13. Sekhar, C.R., Madhu, E., et al.: Mode choice analysis using random forrest decision trees. Transportation Research Procedia **17**, 644–652 (2016)
14. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications **2009**, 1 (2009)
15. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3**(2), 149–175 (1995)
16. Wilson, S.W.: Generalization in the XCS classifier system (1998)
17. Wilson, S.W.: Mining oblique data with XCS. In: International Workshop on Learning Classifier Systems. pp. 158–174. Springer (2000)
18. Wockatz, P., Schartau, P.: Traveller Needs and UK Capability Study. Tech. rep., Transport Systems Catapult (2017)