Learning Qualitative Models by an Autonomous Robot

Jure Žabkar and **Ivan Bratko**
AI Lab, Faculty of Computer
and Information Science,
University of Ljubljana,
SI-1000 Ljubljana, Slovenia

Ashok C Mohan
University of Applied Sciences
Bonn-Rhein-Sieg
Grantham-Allee 20,
53757 Sankt Augustin, Germany

Abstract

In this paper we present a qualitative exploration strategy for an autonomous robot that learns by experimentation. Particularly, we describe a domain in which a mobile robot observes a ball and learns qualitative prediction models from its actions and observation data. At all times it uses these models to predict the results of the actions that it has decided to execute and to design new experiments that would lead it to learn a better model of the world, and for planning of the execution of these experiments. We experimentaly evaluate the exploration strategy.

Introduction

The idea of autonomous robots that are capable of learning by themselves, without any human intervention is one of the most fundamental goals of AI. Among several paradigms of learning, learning by experimentation demands no teacher, but rather learns autonomously, interacting with the real world. In this paper we present a showcase in which an autonomous robot is learning qualitative models by conducting experiments in its environment.

There are several ways of how the robot chooses its actions, designs and plans experiments. In order to learn efficiently, the strategy which it uses to explore its environment is very important. We propose a qualitative exploration strategy for autonomous robot learning. We evaluate our strategy by comparing it to random strategy. The results show that using our strategy, the robot is learning faster and it learns better models. We consider learning of *qualitative* models an important aspect. Qualitative models are easier to learn and sufficient to design and plan the experiment. They reduce the complexity of numerical models considerably and also enable humans to easily understand what the robot has learned.

The robot has no prior knowledge about its environment. In particular, it has no knowledge regarding the relations between its actions and observations. Its task is collecting the data and gradually learning a model which it immediately uses for moving and designing new experiments. Its goal is to learn a model that would relate its actions to its observations. At each step, the robot decides on one of several possible actions. It then uses its current model to predict the result of its action, executes the action and collects observations. It then compares its prediction with the actual observations. The result of this comparison leads to further experiments that helps to revise the model.

In the setting just described, we apply a new method, called parametric Padé, for learning qualitative models in dynamic domains. We present the method itself elsewhere but we provide a short description of the method in section "Parametric Padé" to keep this paper self contained.

Learning qualitative models by experimentation

Our task is to equip the robot with an exploration strategy that would enable the robot to learn without any external intervention. Further, we want the robot to learn qualitative models so that its insights would be easily comprehensible to humans. The robot is restricted to learn by experimentation and to use self generated models for moving and designing new experiments. The robot's motivation for learning is a part of the built-in algorithm. The idea is simple - since the robot depends on its own model, the robot wants to optimize the model's prediction accuracy. To improve the model in time requires collecting new observation data, particularly data most useful for the improvement of the model. Hopefully, if all goes well, after some time the robot will come up with a model whose predictions are always correct, i.e. the robot has learned everything about its actions and their effects in the given environment.

Experimental domain

Our problem domain consists of a mobile robot, a ball and an overview camera, as shown in Fig. 1. The robot uses the overview camera to observe its distance to the ball (ball distance, denoted by bd) and the angle between its orientation and the ball (ball angle, denoted by ba).

The robot is of differential drive type and moves by setting the speeds of the left and the right wheel (L and R respectively). In our case, L and R are always positive, and the robot was restricted to choose between speeds 4 and 5 only. So the robot can move straight ahead ($L = R = 5$), right ($L = 5, R = 4$) and left ($L = 4, R = 5$), as shown in Fig. 2. The robot is not aware of any coordinate system. It

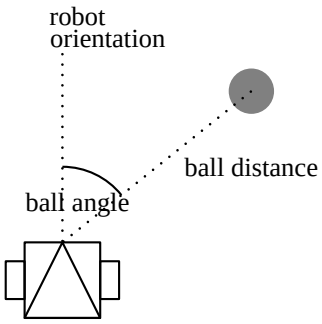


Figure 1: The robot and the ball.

is only aware of the actions it performs (L and R) and the observations from the sensors (bd and ba).

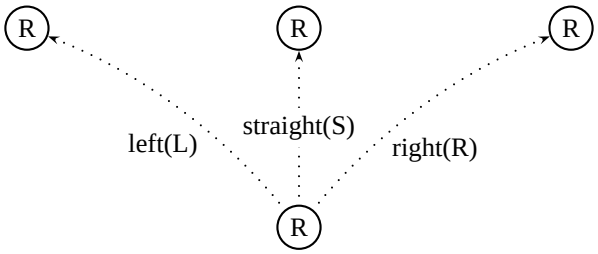


Figure 2: The actions of the robot.

The overall goal that we want the robot to achieve is that it learns a qualitative model describing the relations between its actions and observations. By densely sampling the whole space of above mentioned variables and learning a qualitative tree we obtained an "almost ideal" model of our domain. Note, that we did not use this model in any other way than to see for ourselves what the robot should eventually learn. This "almost ideal" qualitative tree for our domain is shown in Fig. 5. The model is explained in the next section.

We have performed all the experiments in the simulator Simon which is a part of the machine learning framework Orange (Zupan, Leban, & Demšar 2004).

Exploration algorithm

At the begining, i.e. at time t_0, the robot has no knowledge about the effects of its actions. For example, it does not know how its actions from the current state influence its observations in the next step. Namely, there are no relations known to the robot between actions (L and R) and observations (ba and bd).

Without a model the robot can only move by applying random actions, i.e. randomly choose the speed of each wheel. Doing so it collects some data and after a certain time period it learns from the collected data. The learning is supervised. The attributes are robot's actions and observations. The class variable is defined by qualitative relations (described later) between the actions and observations.

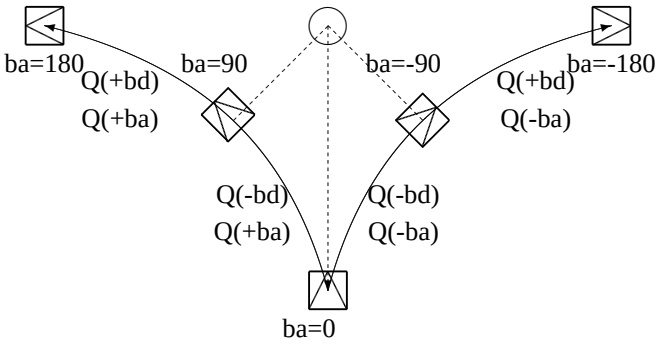


Figure 3: The various angles when robot is turning left and right from $ba = 0$

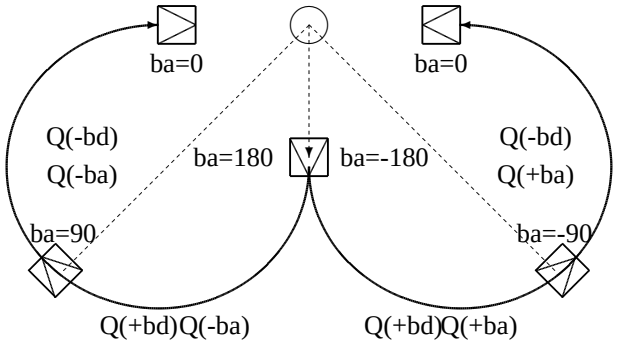


Figure 4: The various angles when robot is turning left and right from $ba = 180$ or $ba = -180$

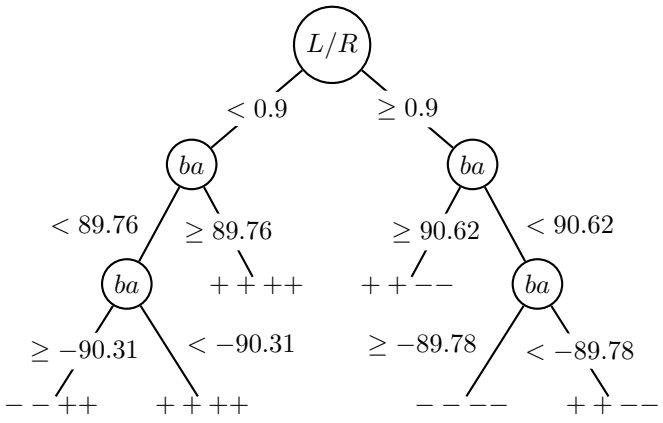


Figure 5: The "almost ideal" model of the robot in our domain.

The robot learns its first model from a small dataset collected by random movement. This initial model is not very accurate and useful. Nevertheless, it enables the robot to use it for making predictions about further actions.

The ability to make predictions enables the choice among learning strategies. A learning strategy determines the next action. The most primitive learning strategy is random strategy, in which the robot chooses one of its three possible actions at random. Random movement is thus defined by actions rather than by robot's positions. The latter is not even possible since in our case the robot is not aware of its coordinates and can not choose to navigate in any coordinate system.

The robot is supposed to learn the relations between its actions and observations. In our simple example, the robot has two actions (L and R) and two observation variables (ba and bd), so it should learn $bd = Q(sS_L)$, $bd = Q(sS_R)$, $ba = Q(sS_L)$ and $ba = Q(sS_R)$, where sign s is $+$ or $-$ and $L = \dot{S}_L, R = \dot{S}_R$, where S_L and S_R are the paths of the left and the right wheel respectively. In these equations, Q stands for qualitative relation as described in section Parametric Padé. In the paper, we use a shorter notation, e.g. "$++--$", giving only the signs s in the above mentioned order. So "$++--$" means: $bd = Q(+S_L)$, $bd = Q(+S_R)$, $ba = Q(-S_L)$ and $ba = Q(-S_R)$. In words: ball distance is increasing when S_L and S_R are increasing (i.e. $L, R > 0$), and ball angle is decreasing when S_L and S_R are increasing. We define the class C of this domain as a 4-tuple of signs as just described. Figures 3 and 4 clearly shows the regions of different values of class C.

Qualitative models that the robot is learning are in the form of qualitative trees (qtree) and qualitative nondeterministic finite automata (envisionment). The robot uses algorithm pPadé with decision trees to learn qualitative trees while it builds an envisionment from the temporal sequence of its actions and observations. The initial set of attributes includes L, R, ba, bd and the class C. To this set, pPadé adds a newly constructed attribute L/R, obtained by the chain rule, dividing the derivatives of each wheel's path w.r.t. time. The attribute L/R describes the qualitative relation between both speeds and can, as we shall see, explain the left and right turns. Using the chain rule for attribute construction is a general principle and is not specifically added to this domain.

There is no relation between the qualitative tree and the envisionment. They are merely a different perspective to the same data. While the qualitative tree is used for prediction, the envisionment is used for planning new experiments for the robot to explore new and less explored regions. Similar to a qualitative tree, the envisionment is gradually learned by the robot.

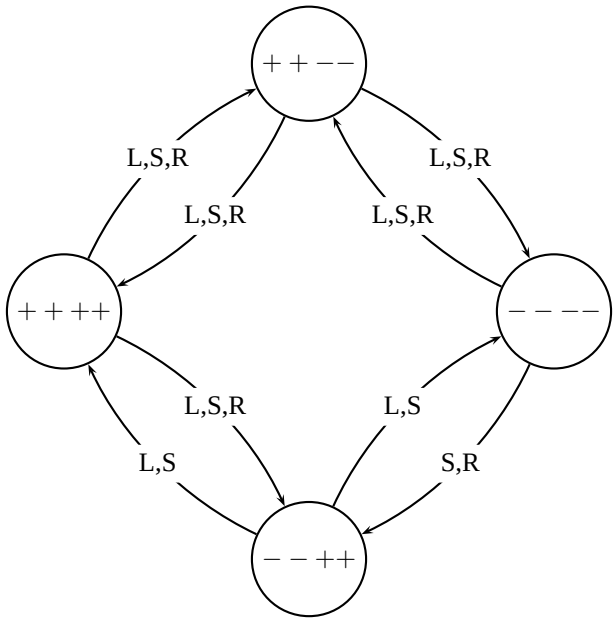


Figure 6: The envisionment learned by the robot.

Exploration strategies

The robot's exploration algorithm includes three strategies that strive to guide the learning towards the final goal. First and most primitive is the *random strategy*. Using this strategy, the robot moves randomly choosing the actions from its set of available actions. The second strategy we call *uniform strategy*; it is used when the robot wants to sample the actions so that their distribution is uniform. Uniform distribution of actions assures that the robot is not biased towards one of the actions, e.g. going straight ahead all the time. At first glance it may seem that uniform and random strategies are the same, but the difference lies in the fact that uniform strategy also accounts for the action executed using persistent strategy. The third strategy is called *persistent strategy*. The robot, using this strategy, keeps executing the same action for some time. Doing so it is collecting more learning examples of the same kind.

The robot uses random strategy only for its first ten moves when it has no knowledge about its environment and the random choice is the best it can make. After it collects the first ten learning examples it can already build a first model and

start using it. At this time, it changes the strategy to *uniform* and enters the main loop in which it is updating and improving the model.

The main loop starts with choosing the next action based on the current strategy (either uniform or persistent). After the robot picks the action it uses the current qualitative tree to make the prediction using the current state and the action. When it makes the prediction it executes the action and observes the result. It compares its own prediction with the actual observation. If they match, the robot continues with persistent strategy, otherwise the robot is "surprised" and motivated for further exploration of the unknown behaviors. The reason for the mismatch is the false prediction of qualitative behavior, i.e. the signs in the class value were predicted wrongly. The robot updates the envisionment with a new state and transition and also updates the qualitative tree. After it updates the model, the robot starts designing a new experiment and planning its actions so that it could carry out the designed experiment. For this purpose it maintains a frequency table of class values and it observes the difference between the number of examples in the current envisionment state and the one with the lowest frequency in the table. If the number of examples in the current envisionment state is greater than a threshold, it selects uniform strategy and picks persistent otherwise. This finishes one iteration of the loop and starts a new one.

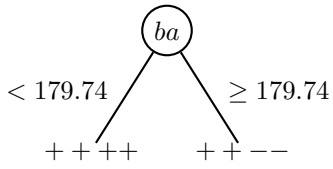


Figure 7: The model created by the robot after 19 steps.

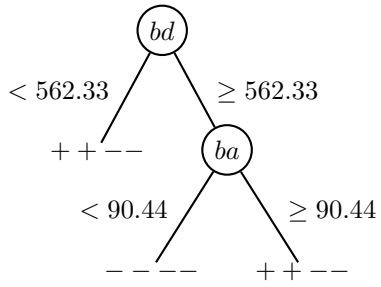


Figure 8: The model created by the robot after 1000 steps.

Results

The exploration algorithm from the previous section enables the robot to learn by experimentation in an efficient way. To confirm the latter, in this section we compare our approach to the pure random strategy. Again, we stress that random strategy does not mean random sampling of the coordinate space but rather choosing the actions randomly.

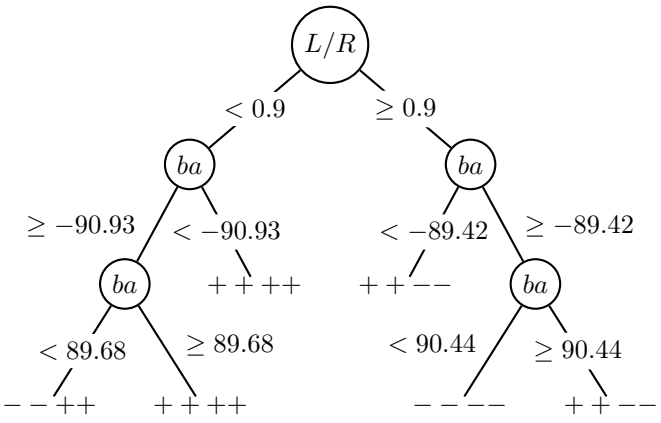


Figure 9: The final model created by the robot after 2674 steps.

In random strategy we use a parameter *duration* which defines the frequency for choosing a random action. If *duration* $= 1$ the action is chosen randomly on each simulation step while for *duration* $= n$ it is chosen only each n-th step and maintained the same in between. The latter is actually not a pure random strategy but rather a mixture of random and persistent. We use it for comparison anyway since the pure random strategy performs extremely poor.

We ran 3 runs of each random strategy, varying *duration* and 9 runs with different initial positions of the robot with our exploration algorithm. We manually determined the point at which the robot learned the desired model. We measured the time it had needed to learn the model in the number of steps it performed until that state. Table 1 presents the results over different runs, the averages and standard errors.

The results show that the robot learns significantly better and faster with our exploration algorithm as opposed to the pure random strategy or random-persistent strategies. We have no formal proof to explain why persistent strategy works. Nevertheless, it is clear from the way humans experiment that we pursue one direction until there arises a reason or motivation to change it.

Parametric Padé

Algorithm Padé, as described in (Žabkar, Bratko, & Demšar 2007), discovers monotonic relations in static domains. It does so by computing partial derivatives from numerical data and is used together with an appropriate machine learning algorithm, e.g. decision trees, to build a qualitative model. However, it is quite limited in the diversity of the domain types it can handle. For example, it can not handle a temporal data set well. Here, we introduce a motivation for a complementary method which we call parametric Padé, abbreviated pPadé. The parameter in pPadé is time which allows pPadé to learn in dynamic domains. We should note here that Padé also works with other parametrizations but time. We only give here time as an example.

Time is not always an important attribute. For example, it is always true that "the larger the piece of ice, the heavier

	Random		our exploration strategy	
Run	Stepsize	Steps taken to reach best model	Stepsize	Steps taken to reach best model
1	1	Not until 30000	1	2674
2		Not until 30000	1	3685
3		Not until 30000	1	1991
4	10	Not until 30000	1	2078
5		Not until 30000	1	3530
6		Not until 30000	1	3254
7	100	15967[a]	1	7317
8		Not until 30000	1	4866
9		27654[b]	1	2843

[a] Even this does not result in the ideal model, but very close to it

[b] This resulted in a model separated at the root by L instead of L/R

Table 1: Comparison between random action selection and our exploration strategy presented here.

it is". In Padé's notation this qualitative proportionality is written as *weight = Q(+volume)*. However, a lot of things change over time and for these time obviously is important. Yet, it should not be treated as any other attribute but rather as a parameter, i.e. the temporal dimension is somehow hidden. For example, it is well known that parametric equations $x(t) = \cos(t)$, $y(t) = \sin(t)$ represent a unit circle for $t \in [0, 2\pi]$. While we observe the circle in xy-plane, parameter t remains hidden. Derivatives w.r.t. time can take advantage of the chain rule:

$$\frac{dy}{dx} = \frac{dy}{dt}\frac{dt}{dx} = \frac{\dot{y}}{\dot{x}}$$

In temporal data sets, it is possible to compute the derivatives of the attributes w.r.t. time t and by using the chain rule, obtain the derivatives w.r.t. other variables as well. Doing so, we overcome the problem of high dimensionality. As opposed to ordinary Padé, where the derivatives are computed in the space of dimensionality n ($n =$ number of attributes), all the derivatives in parametric Padé are computed w.r.t. time.

The input for pPadé is a temporal data set, e.g. a set of points in xy-space (Fig. 10(a)) each having a time stamp. The first four columns of Table 2 present the example data set which we use to illustrate how pPadé works.

The goal in this toy example is to obtain the qualitative behavior of the class variable c w.r.t. attribute x.

The temporal diagram of attributes x and y is shown in Fig. 10(b). First, pPadé computes $\dot{x}$, $\dot{y}$ and $\dot{c}$. pPadé approximates the derivative $\dot{x}$ at t_i as:

$$\dot{x}_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i}.$$

Simple divided differences can be substituted with more robust, noise resistant linear regression, localy weighted regression (LWR) (Atkeson, Moore, & Schaal 1997) or LOESS (Cleveland 1979; Cleveland & Devlin 1988). However, a machine learning algorithm that is subsequently applied to these approximations also tends to eliminate noise.

pPadé uses the chain rule to compute dc/dx as $\dot{c}/\dot{x}$ and similarly of dc/dy. Table 2 presents the computed derivatives and qualitative behavior of c w.r.t. attributes x and y. The signs of dc/dx are also shown in Fig. 10(c). On the other hand, Figure 10(d) shows why it is not possible to correctly asses the desired derivatives in xy-plane, namely the points' neighbors do not respect the time but rather the Euclidean distance in the plane alone.

Related work

The problem we tackled in this paper is addressed in many different research fields which include but are not limited to robotics, AI, psychology and cognitive sciences. We only mention those that are directly related to model building.

Similar to our approach, (Modayil & Kuipers 2007) present an algorithm for learning qualitative models from robot's actions and observations, but their qualitative models are in the form of object control laws while we use qualitative trees and envisionment. An interesting approach using probability estimates is described in (Hart, Grupen, & Jensen 2005). Work by (Barto, Singh, & Chentanez 2004) in intrinsically motivated learning shows how reusable rules can be learned, but only in a playroom domain with much more data than we require. (Kuipers *et al.* 2006) describes a methodology that bootstraps knowledge from low-level sensorimotor primitives and then uses this knowledge to navigate in its environment. (Stoytchev 2005) proposes a novel approach to representing and learning tool affordances by a robot by pushing objects, but with very limited and specific exploratory behaviors.

Conclusion and further work

We showed a simple example of a robot that is capable of learning by making experiments in its environment. The exploration algorithm that we presented proved to be a useful tool for the autonomous learner that has to design, plan and execute the experiments in order to obtain some knowledge about how its actions influence its observations in the given world. One of the contributions in our opinion is the use of qualitative models only and the combination of qualitative tree and the envisionment. Both models do not only suffice to support the robot in its actions, but also offer insights into the knowledge that the robot acquired in the learning

process. Further, we believe that our approach can be generalized to other more complex domains and that it can scale well due to the simplicity of learning the qualitative models.

The algorithm for autonomous learning can be further improved by elaborating the planning part and the design of experiments. Applying this procedure in other domains and with real robots may give rise to new ideas for further development. We are already very close to running a real robot with this algorithm.

Acknowledgment

The work described in this article has been funded by the European Commission's Sixth Framework Programme under contract no. 029427 as part of the Specific Targeted Research Project XPERO ("Robotic Learning by Experimentation").

References

Atkeson, C.; Moore, A.; and Schaal, S. 1997. Locally weighted learning. *Artificial Intelligence Review* 11:11–73.

Barto, A. G.; Singh, S.; and Chentanez, N. 2004. Intrinsically motivated learning of hierarchical collections of skills. *International Conference on Developmental Learning*.

Cleveland, W., and Devlin, S. 1988. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association* 83:596–610.

Cleveland, W. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74:829–836.

Hart, S.; Grupen, R.; and Jensen, D. 2005. A relational representation for procedural task knowledge. In *Proc. 20th National Conf. on Artificial Intelligence*.

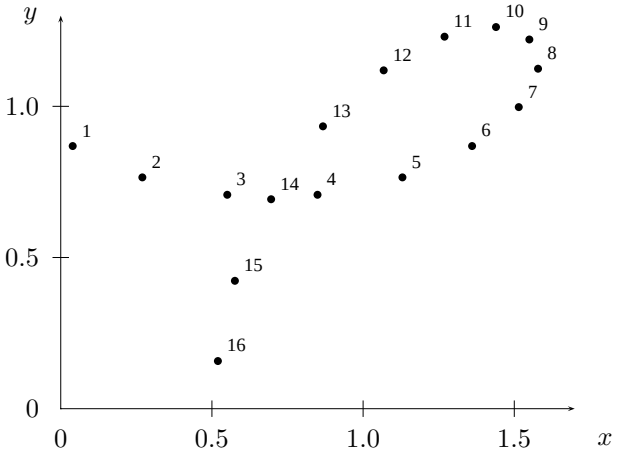
Kuipers, B.; Beeson, P.; Modayil, J.; and Provost, J. 2006. Bootstrap learning of foundational representations.

Modayil, J., and Kuipers, B. 2007. Where do actions come from? autonomous robot learning of objects and actions. *AAAI Spring Symposium Series 2007, Control Mechanisms for Spatial Knowledge Processing in Cognitive / Intelligent Systems*.

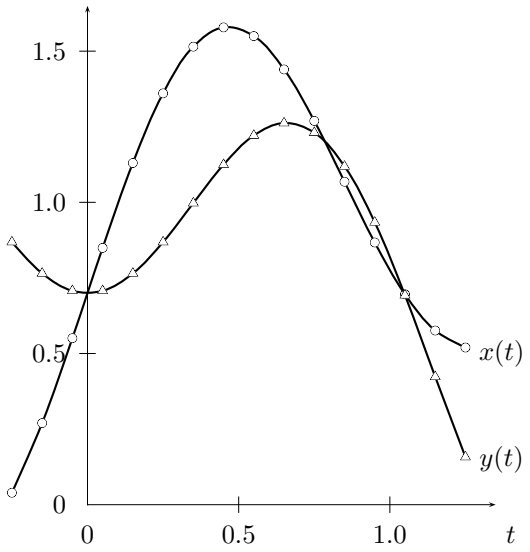
Stoytchev, A. 2005. Behavior-grounded representation of tool affordances. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Žabkar, J.; Bratko, I.; and Demšar, J. 2007. Learning qualitative models through partial derivatives by pad. In *Proceedings of the 21th International Workshop on Qualitative Reasoning*.

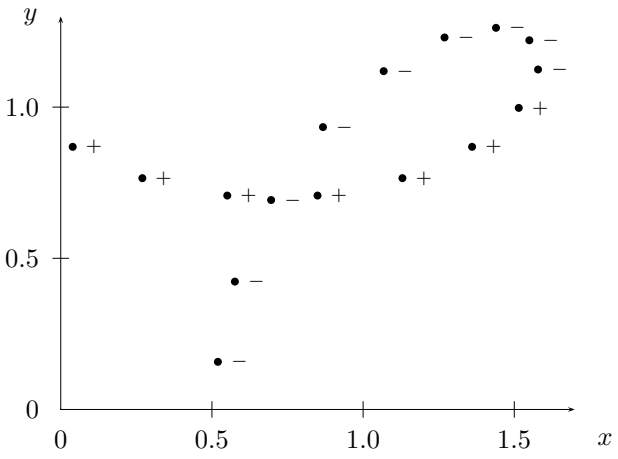
Zupan, B.; Leban, G.; and Demšar, J. 2004. Orange: Widgets and visual programming, a white paper.



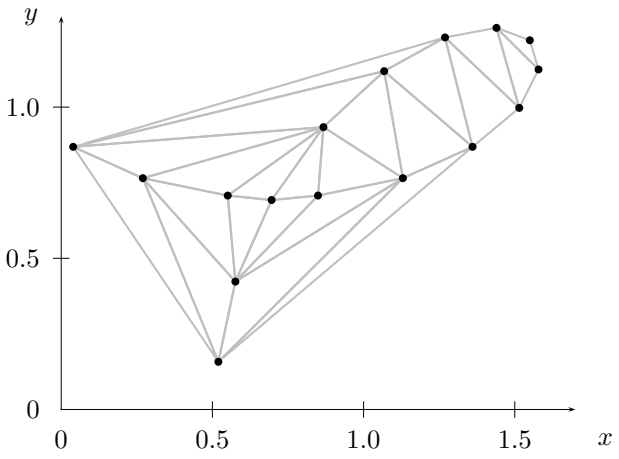
(a) Sampled parametric curve: $x(t) = \sin(3t)\cos(t) + .7$, $y(t) = \sin(3t)\sin(t) + .7$, class variable c takes the values at specified points from $1, \ldots, 16$.



(b) Attributes x and y as functions of t.



(c) Derivatives' signs of dc/dx for each point considering time, calculated using parametric Padé.



(d) Delaunay triangulation of the attribute space. Each point's neighbors could be defined by connections in this graph. So defined neighbors are not good for calculating the derivatives because real neighbors are implied by time.

Figure 10: Illustration of parametric Padé.

t	x	y	c	$\dot{x}$	$\dot{y}$	$\dot{c}$	$\dot{c}/\dot{x}$	$\dot{c}/\dot{y}$	$\dot{y}/\dot{x}$	$c = Q(x)$	$c = Q(y)$
-0.25	0.039	0.868	1	2.30	-1.03	10	4.34	-9.64	-0.44	+	-
-0.15	0.269	0.765	2	2.80	-0.57	10	3.56	-17.38	-0.20	+	-
-0.05	0.550	0.707	3	2.98	0	10	3.35	∞	0	+	o
0.05	0.849	0.707	4	2.80	0.57	10	3.56	17.38	0.20	+	+
0.15	1.130	0.765	5	2.30	1.03	10	4.34	9.64	0.44	+	+
0.25	1.360	0.868	6	1.54	1.28	10	6.47	7.76	0.83	+	+
0.35	1.514	0.997	7	0.63	1.26	10	15.68	7.87	1.99	+	+
0.45	1.578	1.124	8	-0.28	0.96	10	-34.79	10.34	-3.36	-	+
0.55	1.549	1.221	9	-1.10	0.41	10	-9.06	24.30	-0.37	-	+
0.65	1.439	1.262	10	-1.70	-0.31	10	-5.87	-31.41	0.18	-	-
0.75	1.269	1.230	11	-2.01	-1.11	10	-4.96	-8.97	0.55	-	-
0.85	1.068	1.118	12	-2.00	-1.85	10	-4.97	-5.40	0.92	-	-
0.95	0.867	0.933	13	-1.71	-2.41	10	-5.83	-4.14	1.40	-	-
1.05	0.695	0.692	14	-1.19	-2.69	10	-8.34	-3.70	2.25	-	-
1.15	0.576	0.422	15	-0.56	-2.65	10	-17.78	-3.76	4.71	-	-
1.25	0.519	0.157	16	-0.56	-2.65	10	-17.78	-3.76	4.71	-	-

Table 2: The input data (columns 1-4) and the output of parametric Padé (columns 5-12).