Discovering Dynamics

Sašo Džeroski and Ljupčo Todorovski
Jožef Stefan Institute
Jamova 39, 61111 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si Ljupco.Todorovski@ijs.si

Abstract

Machine discovery is concerned with the task of finding laws from experimental and/or observational data. Existing machine discovery systems have mostly generated laws describing static situations. The paper presents LAGRANGE, a system that constructs a set of differential and/or algebraic equations that describe an observed behavior of a dynamic system. As such, LAGRANGE extends the scope of machine discovery to dynamic systems. We show that LAGRANGE is able to generate appropriate sets of laws for several nonlinear dynamic systems from traces of their behavior.

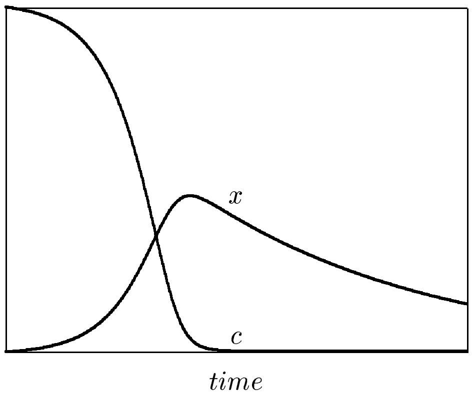

Figure 1: Behavior of a Simple Biological System.

1 Introduction

Consider a simple biological experiment. We place some nutrient and some bacteria in a jar of water. Having done this, we keep the water temperature constant and observe (at regular time intervals) how the concentrations of food and bacteria in the water change over time. Their behavior might look as shown in Figure 1, where c denotes the nutrient concentration and x denotes the bacteria concentration.

Typically, the task of a scientist would be to construct a model of the process that takes place in the jar. Various kinds of models may be constructed, but by far the most common are differential equations. The behavior of the above biological system can be described by the Monod equations [Jorgensen and Johnsen 1989] (p. 301):

$$\dot{c} = \frac{\mu_{max}}{y} \frac{c}{c + k_s} x$$
$$\dot{x} = (\mu_{max} \frac{c}{c + k_s} - k_d) x$$

The system variables of the dynamic system are c (nutrient concentration) and x (bacteria concentration). We define the constants to be $\mu_{max} = 0.1$, $k_d = 0.01$, $k_s = 100$, and $y = 0.6$. The above differential equations, combined with the initial state $c_0 = 300$ and $x_0 = 2$, predict the behavior shown in Figure 1.

One would expect a machine discovery system to help the scientist construct a model of the kind illustrated above, or even construct such a model itself from the observed behavior, and then present it to the scientist for inspection and evaluation. Unfortunately, a typical state-of-the-art machine discovery system is not capable of doing this kind of job. It might try to find a single formula relating c and x directly, disregarding their previous values. In addition, instead of being content with the data given to it and trying to make as much of it as possible, it might ask us to perform additional experiments. If more than two variables were measured, it might ask us to keep the values of all but two variables constant and vary the remaining two. This is inconvenient when we are dealing with dynamic systems, as we might not have experimental control over all the system variables.

The paper presents LAGRANGE, a system that extends the scope of machine discovery to dynamic systems. LAGRANGE is able to find a set of differential and/or algebraic equations, i.e. laws, governing the behavior of a dynamic system, such as the one described in the above example. The task of identification of dynamic systems and the LAGRANGE algorithm, designed to address this task, are described in Section 2. The results of several experiments with LAGRANGE are given in Section 3. Related work is briefly touched upon in Section 4. Finally, Section 5 concludes with a discussion and several directions for further work.

2 The LAGRANGE Algorithm

The task of identification of dynamic systems, addressed by LAGRANGE, can be defined as follows: Given an example behavior of a dynamic system, find a set of laws that describe the dynamics of the system. More precisely, a set of real-valued system variables is measured at regular intervals over a period of time, as illustrated in Table 1. The laws to be discovered (also called a model of the dynamic system) typically take the form of a set of differential equations.

Table 1: A Behavior Trace of a Dynamic System.

time	X_1	X_2	...	X_n
t_0	x_{10}	x_{20}		x_{n0}
$t_0 + h$	x_{11}	x_{21}		x_{n1}
		...		
$t_0 + Nh$	x_{1N}	x_{2N}		x_{nN}

The input to LAGRANGE is a behavior trace of a dynamic system, such as the one given in Table 1. In addition, the values of three parameters have to be specified. These include o, the order of the dynamic system (i.e. the order of the highest derivative appearing in the dynamics equations), d, the maximum depth of new terms introduced by combining old terms (variables), and r, the maximum number of independent regression variables used for generating equations.

Table 2 gives the LAGRANGE algorithm. Taking the set of system variables $S = \{X_1, \ldots, X_n\}$, LAGRANGE first introduces their time derivatives (up to order o). It then introduces new variables (terms) by repeatedly applying multiplication to variables from S and their time derivatives. Finally, given the set of all (old and new) variables, it generates and tests equations by using linear regression.

Table 2: The LAGRANGE Algorithm.

1. Introduce time derivatives (up to order o) of the system variables
   $D := \emptyset$
   for all variables v in S **do**
       $v_0 := v$
       for $i := 1$ **to** o **do**
           $v_i := \frac{d}{dt} v_{i-1}$ (* $v_i = \dot{v}_{i-1}$ *)
           $D := D \cup \{v_i\}$
   $V := S \cup D$

2. Introduce new variables with multiplication
   $V_0 := V$
   for $l := 1$ **to** $d - 1$ **do**
       $V_l := \emptyset$
       for all pairs of variables $(v, u) \in V_0 \times V_{l-1}$ **do**
           $n_{v,u} := v * u$
           $V_l := V_l \cup \{n_{v,u}\}$
           (* duplicate terms removed in this step *)
       $V := V \cup V_l$

3. Do linear regression (LR) on subsets of all variables
   $M := \emptyset$
   for $i := 1$ **to** $r + 1$ **do**
       for all subsets $R \in \mathcal{P}(V)$, $|R| = i$ **do**
           choose dependent variable $y \in R$
           if $LR(y, R \setminus \{y\})$ is significant
           then $M := M \cup \{y = c_0 + \sum_{x \in R \setminus \{y\}} c_x x\}$
           (* add linear regression formula to M *)

The time derivatives are introduced by numerical derivation in step 1. The following formula [Bohte 1991] (p. 73) is used to calculate time derivatives:

$$\dot{x_i} = \frac{1}{12 \cdot h}(x_{i-2} - 8 \cdot x_{i-1} + 8 \cdot x_{i+1} - x_{i+2})$$

where $x_i = X(t_0 + ih)$ and $\dot{x_i} = \dot{X}(t_0 + ih)$. If the measurement error for X is δ and $h < 1$, the error of the numerical derivative calculated by the above formula is $\mathcal{O}(\delta/h)$. The error of the i-th derivative of X calculated in step 1 would then be $\mathcal{O}(\delta/h^i)$. Thus, due care should be exercised and derivatives should be measured whenever the measurement error is lower than the calculation error stated above.

Step 2 introduces as new variables all the terms of depth not greater than d, obtained from the system variables and their derivatives (variables from V_0).

When introducing new variables, terms of depth l are gathered in V_{l-1}. A term $\prod_{i=1}^{n} X_i^{\alpha_i}$ is of depth l iff $l = \sum_{i=1}^{n} \alpha_i$. For example, X_1X_2 is of depth 2 and $X_1^3X_2^2$ is of depth 5. As V_0 contains $(o+1) \cdot n$ variables, V_l contains $\mathcal{O}(((o+1)n)^{l+1})$ variables. Consequently, at the end of step 2, $|V| = \mathcal{O}(((o+1)n)^d)$. The values of the new variables in all time points are also calculated in this step.

Equations are generated and tested in step 3. Roughly speaking, each subset of V sized at most $r+1$ is used to generate a linear equation, where one of the terms is expressed as a linear combination of the remaining ones. The constant coefficients in the linear equation are determined by applying linear regression to fit the corresponding values [Volk 1958] (pp. 260–278). The multiple correlation coefficient R [Volk 1958] (p. 275) is used to judge the significance of the equation. If $R > 1 - t_R$, where t_R is a prespecified threshold (a user definable parameter in LAGRANGE), the equation is considered significant and is retained in the model of the dynamic system. Note that LAGRANGE makes no explicit distinction between dependent and independent variables.

The total number of regressions tried is $\mathcal{O}(|V|^{r+1})$, that is $\mathcal{O}(((o+1)n)^{d(r+1)})$. While this number is exponential in the parameters d and r, we should note that small values of these parameters were sufficient for all the experiments we performed ($d=2$, $r=3$).

To illustrate the work of the algorithm, consider the biological experiment from the introduction, where the system variables are $S = \{c, x\}$. Assume that the values for o, d and r are one, two, and three respectively. The time derivatives $D = \{\dot{c}, \dot{x}\}$ are introduced in the first, and the new terms $V_1 = \{c^2, cx, c\dot{c}, c\dot{x}, x^2, x\dot{c}, x\dot{x}, \dot{c}^2, \dot{c}\dot{x}, \dot{x}^2\}$ in the second step. The third step considers all subsets of $V = S \cup D \cup V_1$ of cardinality one, two, three, and four during equation generation. These include the subsets $\{\dot{c}\}$, $\{\dot{c}, x, \dot{x}\}$ and $\{c\dot{x}, x, \dot{x}, cx\}$, the latter two generating the model equations.

The above description of the LAGRANGE algorithm is declarative. In practice, steps 2 and 3 are interleaved. First, linear equations are tried on the initial set of variables, determined in step 1. New terms are then introduced, layer by layer, and linear regressions for them are tried. This is repeated until the prespecified depth of new terms is reached. In addition, when a term is expressed as a linear combination of others, it is removed from the process of introducing new variables and testing regressions. This reduces the number of redundant equations generated and improves efficiency.

3 Experimental evaluation

The following procedure was used in the experimental evaluation of LAGRANGE: A set of differential equations modeling a real-life dynamic system was first chosen, as well as appropriate values for the parameters involved. The initial state for the system variables was next selected. The differential equations were then integrated for $N = 1000$ steps of $h = 0.01$ time units, using the the fourth-order Runge-Kutta method [Press et al. 1986] (pp. 550-554). The resulting behavior (a sequence of a thousand states) was then given to LAGRANGE, which generated a set of equations describing the behavior.

Models of several real-life dynamic systems were used to test LAGRANGE. These include the biological system described in the introduction, a cascaded tanks system [Kraan et al. 1991], a linear chemical reaction, a predator-prey system, and a nonlinear oscillatory chemical reaction (the Brusselator), the models for all three taken from [Babloyantz 1986]. All of these systems are first-order systems, i.e., the system variables and their first derivatives suffice for describing the dynamics of these systems.

The following parameter settings were used in LAGRANGE: For all test cases, the dynamic system order o was set to one, and r was set to three (at most three terms can appear on the right-hand side of a linear regression). Note that $r = 2$ suffices for all systems, except for the biological one. The maximum depth d of new terms was set to two and the significance threshold t_R was set to 0.00001. The only exception was made for the Brusselator example, where d was set to three (the term X^2Y that appears in the Brusselator equations is of depth three).

LAGRANGE is implemented in the C programming language. It was run on a Sun SPARC IPC workstation. For all models, the running times were of the order of seconds, except for the Brusselator, where the running time was one minute.

An equation might be re-discovered several times in similar forms. For example, the equation $X + Y = Z$ could be re-discovered as $X^2 + XY = XZ$. To simplify the implementation, we have removed such obvious redundancies manually. In future versions of LAGRANGE, this could be done automatically or care might be taken that such redundancies are not generated at all. More sophisticated redundancies have been left untouched, as in the chemical kinetics example.

Biological model

The differential equations given in the introduction were integrated for $N = 1000$ steps of $h = 0.1$ time units. A large integration step was chosen because the dynamics of this system is relatively slow as compared to the other systems considered. LAGRANGE generated the following model from the given behavior:

$$\begin{aligned} \dot{c} &= -\frac{1}{60}\,x - \frac{10}{6}\,\dot{x} \\ c\dot{x} &= -x - 100 \cdot \dot{x} + 0.09 \cdot cx \end{aligned}$$

This model is equivalent to the one given in the introduction if we take into account the particular values of the constants appearing there.

Cascaded tanks

A cascaded tanks system [Kraan et al. 1991] consists of two tanks (A and B), where water flows from the first into the second. The first tank has a constant inflow. The whole system can be described by the following model

$$\begin{aligned} i_A &= c_1 & & \\ \dot{l}_A &= i_A - o_A & \qquad o_A &= c_2\sqrt{l_A} \\ \dot{l}_B &= o_A - o_B & \qquad o_B &= c_2\sqrt{l_B} \end{aligned}$$

where i_A is the inflow into tank A, o_A and o_B are the outflows from tanks A and B and l_A, l_B are the corresponding water levels. The values $c_1 = 200$ and $c_2 = 13$ were chosen for the constants and the behavior of the system was simulated from the initial state $l_{A0} = 10000, l_{B0} = 0$.

The following set of equations, equivalent to the above model, was generated by LAGRANGE:

$$\begin{aligned} i_A &= 200 & & \\ \dot{l}_A &= 200 - o_A & \qquad 169 \cdot l_A &= {o_A}^2 \\ \dot{l}_B &= o_A - o_B & \qquad 169 \cdot l_B &= {o_B}^2 \end{aligned}$$

Taking into account that $i_A = c_1 = 200$, the equations for $i_A, \dot{l}_A$ and $\dot{l}_B$ are identical to the ones in the original model. The remaining two equations can be obtained by squaring their counterparts in the original model.

Chemical kinetics

The following is a simple chemical kinetics model of a two-step chemical reaction, where substance A is transformed into substance B, which is then transformed into substance C [Babloyantz 1986] (p. 37).

$$\begin{aligned} A &\xrightarrow{k_1} B \\ B &\xrightarrow{k_2} C \end{aligned}$$

The constants k_1 and k_2 are the corresponding reaction rates. The evolution of the concentrations of the substances involved in the reaction is described by the following equations:

$$\begin{aligned} \dot{A} &= -k_1 A \\ \dot{B} &= k_1 A - k_2 B \\ \dot{C} &= k_2 B \end{aligned}$$

Only substance A was initially present ($A_0 = 100$, $B_0 = C_0 = 0$), and the reaction rates were $k_1 = 2$ and $k_2 = 3$. An equivalent model was generated by LAGRANGE:

$$\begin{aligned} \dot{A} &= -2 \cdot A \\ \dot{B} &= 2 \cdot A - 3 \cdot B \\ \dot{C} &= 3 \cdot B \\ (*) \quad C &= 100 - A - B \end{aligned}$$

The equations for $\dot{A}, \dot{B}$ and $\dot{C}$ are obtained by filling in the values of the constants k_1 and k_2 in the original model. Equation $(*)$, on the other hand, is not found in the original model. It expresses the law of mass conservation for the particular case at hand.

Population dynamics

A Volterra-Lotka model of periodic behavior can be used to model the coexistence of prey and predator populations [Babloyantz 1986] (p. 145). For example, take the populations of lynxes and hares. The latter is a vegetarian, the former a carnivore that hunts hares. The lynx population must behave in such a manner that it must not eat all the hares, or its own species will disappear. The dynamics of this system is described by the following model

$$\begin{aligned} \dot{N}_1 &= k_1 N_1 - s N_1 N_2 \\ \dot{N}_2 &= s N_1 N_2 - k_2 N_2 \end{aligned}$$

where N_1 and N_2 represent the hare and lynx populations, respectively. We chose the initial hare and lynx populations to be $N_{10} = 10$ and $N_{20} = 140$, and the parameters $s = 0.01, k_1 = 1.6, k_2 = 0.2$. The following two equations were generated by LAGRANGE:

$$\begin{aligned} N_1 N_2 &= 160 \cdot N_1 - 100 \cdot \dot{N}_1 \\ N_1 N_2 &= 20 \cdot N_2 + 100 \cdot \dot{N}_2 \end{aligned}$$

They can be obtained by substituting the values of the parameters in the original equations, then multiplying by $100 = 1/s$ and expressing $N_1 N_2$ from the resulting equations.

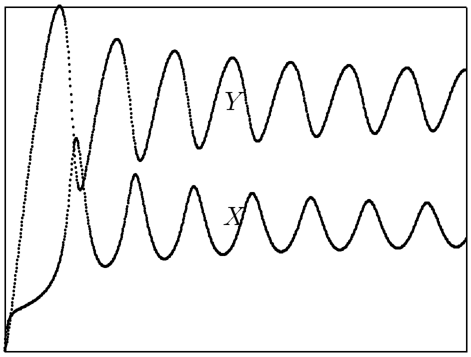

Figure 2: Behavior of the Brusselator.

The Brusselator

The Brusselator is an oscillating chemical system [Babloyantz 1986] (pp. 175–177), in which the concentrations of two substances X and Y are trapped in an oscillatory time change, described by the following equations and illustrated in Figure 2.

$$\begin{aligned}\dot{X} &= A - (B+1)X + X^2Y \\ \dot{Y} &= BX - X^2Y\end{aligned}$$

Substances A and B also take part in the reaction, but their concentrations are held constant through appropriate adjustments. The initial conditions were $X_0 = Y_0 = 0$, and the parameters were $A = 1$ and $B = 2$. From the behavior trace, LAGRANGE generated the following model:

$$\begin{aligned}X^2Y &= -1 + 3 \cdot X + \dot{X} \\ \dot{Y} &= 1 - X - \dot{X}\end{aligned}$$

Expressing the term X^2Y from the original equation for $\dot{X}$ yields the first equation generated by LAGRANGE. The second equation generated by LAGRANGE is obtained by adding the original equations and then expressing $\dot{Y}$ in terms of X and $\dot{X}$.

4 Related work

To relate our work to other work in the field of machine discovery and machine learning, let us briefly discuss the history of LAGRANGE. Although machine discovery has not paid much attention to dynamic systems (except for the recent recognition of the need for techniques that can handle such systems [Bielecki 1992]), there is a significant body of work devoted to learning qualitative models of dynamic systems [Coiera 1989], [Bratko et al. 1991], [Kraan et al. 1991], [Džeroski and Bratko 1992]. Some of this work relies on inductive logic programming [Lavrač and Džeroski 1993] systems to induce the models from example behaviors and background knowledge consisting of the definitions of the QSIM [Kuipers 1986] constraints.

The LINUS approach to inductive logic programming [Lavrač et al. 1991] is based on the idea of transforming an inductive logic programming problem to propositional form and then applying propositional learning systems. This is accomplished by using background knowledge predicates to introduce new variables and generate propositional features [Džeroski et al. 1992]. Our original idea for LAGRANGE was to use a similar approach in order to discover system dynamics: introduce new variables as the time derivatives of existing ones and then apply existing machine discovery systems. However, the latter did not meet our basic requirements for a machine discovery system to be used for discovering differential equations, which were:

- It must be able to find laws involving more than one variable from observational data only, i.e. without asking for additional experiments.
- It must be able to find a set of laws, rather than a single one, where all laws hold for the domain as a whole. It is not explicitly stated beforehand which system variables are dependent and which are independent.

Systems like BACON [Langley et al. 1987] and FAHRENHEIT [Żytkow and Zhu 1991] typically ask the scientist to perform experiments. While ABACUS [Falkenheiner and Michalski 1990] and FAHRENHEIT are able to discover more than one law in a domain, these laws usually hold in disjoint subspaces of the domain. There have been some ideas about discovering a set of simple laws from observational data with BACON, but they have not actually been incorporated into BACON. Thus, none of the above system meets both of the above criteria in a satisfactory way.

The Equation Finder [Zembowitz and Żytkow 1992], used as a module in FAHRENHEIT, comes closest to meeting the above requirements. Given a set of real-valued pairs (x_i, y_i), it tries to find a formula $Y = f(X)$, without asking for additional data. The main problem with Equation Finder is that it is looking for a function of one variable only. In dynamic systems, however, the value of a variable at time t_{i+1} can depend on the values of several variables at time

t_i. This makes the ability to deal with functions of more than one variable a necessity.

One of the important features of Equation Finder is the ability to handle errors in the input data. Such a feature is necessary for handling real-life data. It is also important that the convergence of Equation Finder and its sensitivity to errors in the input data have been thoroughly analysed [Zembowitz and Żytkow 1992].

IDS [Nordhausen and Langley 1990] touches upon the problem of discovering dynamics. The qualitative schemata generated by IDS can be considered qualitative states of a system. The transitions between states (schemata) capture in a way the qualitative change of the system over time, i.e., its global dynamics. They are more of the phase transition type, rather than continuous state changes as described by differential equations. However, numeric laws within the qualitative states could, in principle, include time-dependent variables.

5 Discussion

The most important contribution of our work is the extension of the scope of machine discovery to dynamic systems. LAGRANGE is able to construct a set of differential and/or algebraic equations describing a given behavior of a dynamic system. In this way, it extends the scope of machine discovery systems from high-school physics ($F = ma$) to college physics ($F = m\ddot{x}$).

It is also important that LAGRANGE is able to generate a set of laws involving more than two variables from observational data only. This is in contrast with most machine discovery systems, which perform problem decomposition to find laws involving more than two variables. In practice, these systems keep some variables constant and ask the user (scientist) to vary the others. This may prove impossible in many cases, especially in the context of dynamic systems, as the scientist might not have full experimental control over all the system variables involved.

Our work is very much in the BACON tradition, showing that one can handle a large number of apparently complex laws with very simple discovery mechanisms. This is demonstrated by the successful use of LAGRANGE for generating models of several dynamic systems, including a biological system, a cascaded tanks system, two chemical reactions and a predator-prey population system. For these systems, LAGRANGE generates models correctly describing the given behaviors. Judging on the successful use of LAGRANGE in building models of dynamic systems, we conclude that LAGRANGE is a promising step towards the application of machine discovery to complex dynamic systems.

LAGRANGE can be improved in many ways, including handling of errors in the input data, taking into account multiple behavior traces, removing redundant laws in an intelligent way, introducing new terms by other transformations in addition to multiplication, and improving the search control.

The highest priority is the handling of measurement errors and noise in the input data, which is necessary for practical applications. At present, LAGRANGE does not take into account any errors in the input data, except through the t_R tolerance parameter (Section 2). Namely, if the data is less accurate, we may set t_R to a lower value and consider equations that are apparently less significant. However, a careful analysis of the sensitivity of LAGRANGE to changes in this parameter is needed. For applying LAGRANGE to real-life data, measurement errors should be taken into account more explicitly. The approach in Equation Finder [Zembowitz and Żytkow 1992] could be adapted for this purpose.

Although at present new terms are introduced by multiplication only, other transformations, such as those used in Equation Finder could be easily incorporated. The number of new terms would significantly increase in this case and would require reduction of the search complexity. Heuristics should thus be used to reduce the number of new terms introduced. In addition, the terms for linear regressions should be chosen in a more intelligent way.

Acknowledgements

This work is financially supported by the Slovenian Ministry of Science and Technology and in part by the ESPRIT III Basic Research Project No. 6020 on Inductive Logic Programming. Many thanks to Pat Langley and Jan Żytkow for their thorough and stimulating comments. Thanks also to Boris Kompare for suggesting the biological model as a test case.

References

[Babloyantz 1986]
Babloyantz, A. (1986). *Molecules, Dynamics, and Life.* John Wiley & Sons, Inc., New York.

[Bielecki 1992]
Bielecki, M. W. (1992). Machine discovery approach to dynamic systems in the real laboratory.

In Żytkow, J. M., editor, *Proc. of the ML92 Workshop on Machine Discovery*, pages 58–62.

[Bohte 1991]
Bohte, Z. (1991). *Numerical Methods*. The Society of Mathematicians, Physicists and Astronomers of Slovenia, Ljubljana. In Slovenian.

[Bratko et al. 1991]
Bratko, I., Muggleton, S., and Varšek, A. (1991). Learning qualitative models of dynamic systems. In *Proc. Eighth International Workshop on Machine Learning*, pages 385–388. Morgan Kaufmann, San Mateo, CA.

[Coiera 1989]
Coiera, E. (1989). Learning qualitative models from example behaviours. In *Proc. Third International Workshop on Qualitative Physics*. Stanford, California.

[Džeroski and Bratko 1992]
Džeroski, S. and Bratko, I. (1992). Handling noise in inductive logic programming. In *Proc. Second International Workshop on Inductive Logic Programming*. Tokyo, Japan. ICOT TM-1182.

[Džeroski et al. 1992]
Džeroski, S., Muggleton, S., and Russell, S. (1992). PAC-learnability of determinate logic programs. In *Proc. Fifth ACM Workshop on Computational Learning Theory*, pages 128–135. ACM Press, New York, NY.

[Falkenheiner and Michalski 1990]
Falkenheiner, B. and Michalski, R. (1990). Integrating quantitative and qualitative discovery in the ABACUS system. In Kodratoff, Y. and Michalski, R., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 153–190. Morgan Kaufmann, San Mateo, CA.

[Jorgensen and Johnsen 1989]
Jorgensen, S. and Johnsen, E. (1989). *Principles of Environmental Science and Technology*. Elsevier, Amsterdam.

[Kraan et al. 1991]
Kraan, I., Richards, B., and Kuipers, B. (1991). Automatic abduction of qualitative models. In *Proc. Fifth International Workshop on Qualitative Physics*. Austin, Texas.

[Kuipers 1986]
Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence*, 29(3):289–338.

[Langley et al. 1987]
Langley, P., Simon, H., and Bradshaw, G. (1987). Heuristics for empirical discovery. In Bolc, L., editor, *Computational Models of Learning*. Springer, Berlin.

[Lavrač and Džeroski 1993]
Lavrač, N. and Džeroski, S. (1993). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, Chichester.

[Lavrač et al. 1991]
Lavrač, N., Džeroski, S., and Grobelnik, M. (1991). Learning nonrecursive definitions of relations with LINUS. In *Proc. Fifth European Working Session on Learning*, pages 265–281. Springer, Berlin.

[Nordhausen and Langley 1990]
Nordhausen, B. and Langley, P. (1990). A robust approach to numeric discovery. In *Proc. Seventh International Conference on Machine Learning*, pages 411–418. Morgan Kaufmann, San Mateo, CA.

[Press et al. 1986]
Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical Recipes*. Cambridge University Press, Cambridge, MA.

[Volk 1958]
Volk, W. (1958). *Applied Statistics for Engineers*. McGraw-Hill, New York, NY.

[Zembowitz and Żytkow 1992]
Zembowitz, R. and Żytkow, J. (1992). Discovery of equations: experimental evaluation of convergence. In *Proc. Tenth National Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

[Żytkow and Zhu 1991]
Żytkow, J. and Zhu, J. (1991). Application of empirical discovery in knowledge acquisition. In *Proc. Fifth European Working Session on Learning*, pages 101–117. Springer, Berlin.