Function Discovery using Data Transformation

Thong H. Phan — phant@cpsc.ucalgary.ca

Computer Science Department, University of Calgary, Calgary, Canada

Ian H. Witten — ihw@waikato.ac.nz

Computer Science Department, University of Waikato, Hamilton, New Zealand

1 Introduction

Function discovery is the problem of finding a symbolic formula for a function $f : D_1 \rightarrow D_2$ from a set of examples $\{(x, y) \mid y = f(x)\} \subset D_1 \times D_2$. Acceptable solutions are restricted to formulas expressible in some given description language $\mathcal{L}$. In most previous discovery systems—for example, BACON [4], ABACUS [1], COPER [2], FAHRENHEIT [3] and IDS [5]—the description language is restricted to contain only rational functions so that symbolic descriptions can easily be enumerated. This paper shows how the idea of *data transformation*, a technique pioneered in FFD [6], can be used as the basis of a far more comprehensive description language that includes all functions that can be transformed to rational functions by differentiation and logarithm operations.

The data transformation approach induces functions by searching through a space of transformations that can be expressed both symbolically and as operations on the example set. It expresses the solution implicitly, as a small system of simultaneous equations, some of which may be differential equations. Solving these symbolically would (where possible) give a formula for the unknown function. In order to operationalize transformations while maintaining the ability of the example set to faithfully represent transformed functions, it is necessary to allow for new examples to be requested interactively. The main contribution of this paper is to define a transformation-based description language and characterize its representational power. We also briefly sketch a practical implementation of a function induction system that uses this approach.

Name	Transformation
Function inverse	$Inv : (x, y) \rightarrow (y, x)$
Reciprocal	$R : (x, y) \rightarrow (x, 1/y)$
Logarithm	$L_c : (x, y) \rightarrow (x, \log(c \times y))$
Linear factoring	$F_{(c,y_c)} : (x, y) \rightarrow (x, (y - y_c)/(x - c))$
Differentiation	$D_{(c,y_c)} : (x, y) \rightarrow (x, dy/dx)$

Table 1: Available transformations

2 The function description language

We define a particular function description language $\mathcal{L}$, assumed throughout the remainder of this paper, in which any formula is expressed as a pair of equations of the form:

$$\begin{aligned} \mathtt{T}_k \circ \cdots \circ \mathtt{T}_2 \circ \mathtt{T}_1(x, y) &= (u, v) \\ c_1 v^2 + c_2 uv + c_3 u^2 + c_4 v + c_5 u + c_6 &= 0. \end{aligned}$$

Each $\mathtt{T}_i$ is one of the five transformations defined in Table 1; x, y, u and v are symbolic variables; and $c_1, \ldots, c_6$ are real numbers of which at least one is non-zero. By convention, the variable y denotes the function's value and x is the independent variable. The sequence $\mathtt{T}_k \circ \cdots \circ \mathtt{T}_1$ is called a *transformation sequence* and the formula $c_1 v^2 + c_2 uv + c_3 u^2 + c_4 v + c_5 u + c_6$ is called a *matching pattern*. Since the expressions on the right-hand side of the defining equations are always (u, v) and 0, respectively, the full description above can be abbreviated to a pair ($\mathtt{T}_k \circ \cdots \circ \mathtt{T}_1$, $c_1 v^2 + c_2 uv + c_3 u^2 + c_4 v + c_5 u + c_6$).

In Table 1, the subscripts c of the logarithm and factoring transformations L_c and $F_{(c,y_c)}$ denote a free parameter that must be chosen when applying them. The value y_c in $F_{(c,y_c)}$ represents the function's value at $x = c$, so that, despite appearances, only one free parameter is involved. In the case of $D_{(c,y_c)}$, although the operation itself requires no parameter, (c, y_c) is retained so that the transformation can be inverted to yield a unique result.

As an example, consider the function $y = \log(x^2 + 2) + x + 1$. One expression of it in $\mathcal{L}$ is:

$$\begin{aligned} R \circ F_{(1,\frac{5}{3})} \circ D_{(1,2+\log 3)}(x,y) &= (u,v) \\ uv - 2v + \tfrac{3}{2}u^2 + 3 &= 0. \end{aligned}$$

In short-hand notation, this becomes $(R \circ F_{(1,\frac{5}{3})} \circ D_{(1,2+\log 3)},\ uv - 2v + \frac{3}{2}u^2 + 3)$. The parameters c_1 and c_2 in the transformations $F_{(c_1,y_{c_1})}$ and $D_{(c_2,y_{c_2})}$ are both arbitrarily chosen to be 1, and so $y_{c_1} = \frac{5}{3}$ and $y_{c_2} = 2 + \log 3$ in this example. To show that this description represents the original function, we perform the necessary transformations and solve the resulting system of equations. Applying the sequence of transformations to (x, y) yields

$$R \circ F_{(1,\frac{5}{3})} \circ D_{(1,2+\log 3)}(x,y) = \left(x, \frac{x-1}{dy/dx - \frac{5}{3}}\right).$$

Equating this to (u, v) and eliminating u and v using $uv - 2v + \frac{3}{2}u^2 + 3 = 0$ gives the equation

$$\frac{x-1}{dy/dx - \frac{5}{3}} = \frac{-\frac{3}{2}x^2 - 3}{x-2}.$$

Solving for dy/dx produces the differential equation $\dfrac{dy}{dx} = 1 + \dfrac{2x}{x^2+2}$, which satisfies all functions of the form $y = \log(x^2+2) + x + C$ for any constant C. The subscript in the transformation $D_{(1,2+\log 3)}$ provides an initial condition for this differential equation. In this case, $x = 1$ and $y = 2 + \log 3$ gives $C = 1$.

Note that since each transformation has a unique inverse, function descriptions, in principle, can be converted to corresponding formulas in conventional form. However, because of the inclusion of the function inverse transformation Inv and the differentiation transformation D, many are not expressible in terms of elementary functions. Consequently, if explicit formulas are desired, symbolic mathematical routines may be included to solve the resulting implicit equations whenever possible; however, this is not the concern of the present paper.

We now exhibit some theoretical results that illustrate the expressiveness of the language $\mathcal{L}$ defined above. For convenience, the following notation for sequences of transformations will be used: A sequence $\mathtt{T}_k \circ \cdots \circ \mathtt{T}_1$ of transformations of the same type $\mathtt{T}$ (including the sequence of length zero, $k = 0$) is written $\mathtt{T}^*$. $\{\mathtt{T}, \mathtt{T}'\}^*$ denotes a sequence composed only of transformations of

the types occurring between the brackets {}, in this case $\mathtt{T}$ and $\mathtt{T}'$.

Proposition 1 *Let $y = f(x)$ be a function that can be described in $\mathcal{L}$ as ($\mathtt{T}_k \circ \cdots \circ \mathtt{T}_2 \circ \mathtt{T}_1$, $c_1v^2 + c_2uv + c_3u^2 + c_4v + c_5u + c_6$), where $\mathtt{T}_i$, $1 \leq i \leq k$, are transformations in Table 1, and $c_1, \ldots, c_6$ are real constants of which at least one is non-zero. If a function $\hat{y} = h(\hat{x})$ has the property that $\mathtt{T}_0(\hat{x}, \hat{y}) = (x, y)$ where $\mathtt{T}_0$ is an available transformation, then $\hat{y} = h(\hat{x})$ can be described as ($\mathtt{T}_k \circ \cdots \circ \mathtt{T}_2 \circ \mathtt{T}_1 \circ \mathtt{T}_0$, $c_1v^2 + c_2uv + c_3u^2 + c_4v + c_5u + c_6$).*

PROOF: The proposition follows from the definition of function descriptions in $\mathcal{L}$. □

Proposition 2 *Descriptions of the form $(\{R, F\}^*, v + c)$ describe all and only rational functions.*

PROOF: The proposition follows from two properties that (1) any rational function can be transformed to a constant c by repeated applications of R and F, and (2) repeated applications of the reverse transformations R^{-1} and F^{-1} to a constant c will produce rational functions only. (R^{-1} and F^{-1} are defined as $R^{-1}(x, y) = R(x, y) = (x, 1/y)$ and $F^{-1}_{(c,y_c)} = (x, y * (x - c) + y_c)$.) □

By Proposition 1, any function that can be transformed to a function expressible in $\mathcal{L}$ is also expressible in $\mathcal{L}$. Consequently, we have shown that any function that can be transformed to a rational function using the transformations defined in Table 1 is also expressible in $\mathcal{L}$. In the following propositions, some interesting classes of functions are shown to be expressible in $\mathcal{L}$.

Proposition 3 *Descriptions of the form $(\{R, F\}^* \circ D^*, v + c)$ express functions of the form*

$$y = p_k + \sum_i c_{1,i} x^{n_{1,i}} \log(a_{1,i}x + b_{1,i}) + \sum_i c_{2,i} x^{n_{2,i}} \tan^{-1}(a_{2,i}x + b_{2,i}) + \sum_i \frac{c_{3,i}}{(a_{3,i}x + b_{3,i})^i} + \sum_i \sum_j \frac{a_{4,i,j}x + b_{4,i,j}}{(h_i)^j},$$

where $n_{1,i}$, $n_{2,i}$ are non-negative integers; p_k is a polynomial with degree at most k; each h_i is an irreducible polynomial of degree 2; each of $a_{i,j}$, $a_{4,i,j}$, $b_{i,j}$, $b_{4,i,j}$, and $c_{k,i}$ is a real constant; and each $\sum_i$ represents a finite summation of similar terms.

PROOF: The proposition follows from the two properties: (1) repeated applications of D to a function of the above form will produce a rational function, and (2) repeated integration of any rational function will produce a function of the above form. □

Function form	Transformations	Function form	Transformations
$f(x) = e^{g(x)}$	L	$f(x) = e^{g^{-1}(x)}$	$Inv \circ L$
$f(x) = g(\log(x))$	$Inv \circ L \circ Inv$	$f(x) = g^{-1}(\log x)$	$L \circ Inv$
$f(x) = e^{g(\log(x))}$	$Inv \circ L \circ Inv \circ L$	$f(x) = e^{g^{-1}(\log x)}$	$L \circ Inv \circ L$

Table 2: Transformation sequences for some transcendental functions

Note that functions in Proposition 3 only require a constant matching pattern in their descriptions. When non-constant patterns are used, other functions can also be expressed using sequences of the type $\{R, F\}^* \circ D^*$ as well. (For example, $y = x^{3/2} + x$ can be expressed as $(D_{(0,0)}, v^2 - 2v - \frac{9}{4}u + 1)$.) Furthermore, functions expressible in $\mathcal{L}$ are not necessarily restricted to rational functions or their integrations. Table 2 presents the necessary sequences to transform some transcendental functions to a function g expressible in $\mathcal{L}$. Proposition 4 illustrates two special classes of differential equations that can also be expressed in $\mathcal{L}$.

Proposition 4 *If g is expressible in $\mathcal{L}$, then ordinary differential equations of the form $y' = yg(x)$ and autonomous differential equations of the form $y' = g(y)$ are also expressible in $\mathcal{L}$.*

Proof: ¿From Table 1, $D \circ L(x, y) = (x, \frac{d \log y}{dx}) = (x, y'/y)$, and $R \circ D \circ Inv(x, y) = (y, \frac{1}{dx/dy}) = (y, dy/dx) = (y, y')$. Thus, ordinary differential equations of the form $y' = yg(x)$ can be written as $(\mathcal{T} \circ D \circ L, Q)$, where $(\mathcal{T}, Q)$ is the description of g. Similarly, autonomous differential equations of the form $y' = g(y)$ can be expressed as $(\mathcal{T} \circ R \circ D \circ Inv, Q)$. □

Note that for successful application of $R \circ D \circ Inv$, $f(x)$ must be monotonic. While this condition limits the usefulness of the sequence $R \circ D \circ Inv$, it may be circumvented in practice to provide the result (y, y') directly. The case of $D \circ L$ is similar: the restriction that f must not contain both positive and negative values can be finessed by directly computing the result $(x, y'/y)$. Consequently, with proper implementation, the combined sequences $D \circ L$ and $R \circ D \circ Inv$ can be used to describe functions expressed as differential equations that take the form of rational functions of x and y'/y, or of y and y'. For instance, trigonometric functions of the form $y =$

$c_1 \sin(ax+b) + c_2 \cos(ax+b)$ can be expressed in $\mathcal{L}$ as $(R \circ D \circ Inv,\ a^2u + v^2 - a^2c_1^2 - a^2c_2^2)$ even though most of them are nonmonotonic functions.

3 Data transformation as a discovery technique

Data transformation is the discovery technique used to implement the description language of Section 2. Formally, it can be viewed as an implementation of the converse of Proposition 1:

> *If* $(T_k \circ \ldots \circ T_1,\ Q)$ *is a description of some function* $y = f(x)$*, then* $(T_k \circ \ldots \circ T_2,\ Q)$ *is a description of some function* $\hat{y} = h(\hat{x})$ *where* $(\hat{x}, \hat{y}) = T_1(x, y)$.

To discover a description of some function $y = f(x)$, one can proceed by first discovering a description of the new function $\hat{y} = h(\hat{x})$. Examples $(\hat{x}, \hat{y})$ of h are obtained by applying the transformation T_1 to the examples (x, y) of f. In effect, the problem of discovering f is reduced to that of discovering h, a function whose description has a shorter transformation sequence.

¿From an operational perspective, one searches for a transformation sequence $T_k \circ \ldots \circ T_1$ that transforms examples of the unknown function to examples of a quadratic relation, instead of constructing all function descriptions and solving the corresponding system of equations. Properly implemented, this approach has the practical advantage that numerical transformations and curve-fitting operations can be performed far more cheaply than systems of equations—which include nonlinear and differential equations—can be constructed and solved. However, naive implementations suffer problems of incomplete generation of function descriptions and inexact numerical computations.

- *Incomplete generation of function descriptions*

Recall that a function description comprises two items: a sequence of transformations and a matching pattern. The method proceeds by applying the different sequences of transformations to the set of examples until a matching pattern is found. The problem is that the choice of sequences is constrained by the fact that the transformations Inv and L cannot be applied to all functions. Specifically, inverting a function yields a relation rather than a function. The idea of inverting

$y = f(x)$ only makes sense if the inverse function $x = f^{-1}(y)$ exists. Consequently, Inv cannot be applied unless one is certain that the result is a function. The case of L is analogous; it cannot be applied to functions that have both positive and negative values.

In both cases, the constraints on Inv and L effectively prevent certain examples from being transformed any further. As a result, if the examples are not carefully chosen, some functions may not be discovered using the data transformation technique even though they can be described in $\mathcal{L}$. Thus it appears that the approach is incomplete in the sense that it can fail to discover some descriptions that can be couched in the language under consideration.

- *Inexact computation of transformations*

Of the five transformations in Table 1, the computation of D and F implicitly refers to the function implied by the examples. D computes its derivative and F its ratio with respect to some linear factor. Because the formula for the function implied by the examples is not available, one can only provide numerical approximations to these transformations. To compound the problem, the inexact nature of D and F can easily lead to results that are meaningless. The cumulative error can grow large enough to produce incorrect solutions, or perhaps no solution at all—because an inconsistent system of equations may result from the correct transformation sequence.

To increase the accuracy of such approximations, the standard method is to use examples that are closely spaced in x. However, with the inclusion of Inv, this requirement is impossible to satisfy after some transformations have been applied to the original examples. Furthermore, from a practical point of view, using too many examples consumes computing resources that could be reduced if the approximation errors were estimated more accurately. In short, the standard method is not satisfactory because of the difficulty of ensuring that the examples remain closely spaced even after several transformations, and of estimating in advance the number of examples required to maintain a desired accuracy level.

4 Overview of LINUS

LINUS is a function induction system that implements the data transformation approach. To address the above problems, its design follows three principles: examples must be selectable on demand; all transformations must be numerically applicable to the examples selected; and transformations must be numerically reversible. The first two combine to solve the problem of incomplete generation of function descriptions caused by mathematical constraints on the transformations Inv and L. The third provides a mechanism for estimating the errors incurred by inexact application of the transformations.

LINUS is empowered to select examples that are deemed to be helpful in identifying the unknown function. The idea is that, by examining the examples currently available, it is possible to determine which ones would be best to use in the next operation. If these are not available, a request is issued for them before proceeding further. Before each transformation is applied, the current examples are examined using the error analysis process sketched below to see if the examples are sufficient to support the upcoming transformation. If not, more examples are requested to cover places where the analysis indicates that the error exceeds some acceptable level. Most importantly, if the current examples prevent the application of any transformation, it must be determined which new examples will enable the blocked transformation to proceed, in order that these can be specifically requested. This solves the problem of incomplete generation of function descriptions.

The process of error analysis comprises two parts: the replacement of examples of the unknown function by local approximations, and the estimation of transformation errors. "Local approximations" are piecewise rational functions composed of polynomials of degree 3 or less. Before the transformations D and F are applied, a set of local approximations is computed using least-squares curve-fitting. These approximations are considered reliable if the fitting error is less than some predefined tolerance level. Then, instead of applying D or F to the examples directly, the transformation is applied symbolically to the corresponding local approximations, and the results are evaluated with respect to the examples available.

Transformation errors are estimated from the curve-fitting process. Let (x, y) be the examples before the transformation $\mathtt{T}$ is applied, and $\mathtt{T}(x, y) = (u, v)$ be the result after that transformation. Ideally, we would have $(x, y) = \mathtt{T}^{-1}(u, v)$, but in practice, the result is $\mathtt{T}^{-1}(u, v) = (x + \epsilon_x, y + \epsilon_y)$, where ϵ_x and ϵ_y are the accumulated errors of both the transformation $\mathtt{T}$ and its inverse $\mathtt{T}^{-1}$. If both errors fall within the expected fitting error, the result of the transformation is considered reliable. Otherwise, the local approximations are not sufficiently accurate for the transformation, and more examples are requested where needed.

Once a description is identified as a potential solution, LINUS verifies that it represents a function that is numerically indistinguishable from the unknown one within a pre-specified tolerance. First, the precision of the parameters in the transformation sequence is improved. To achieve this, more closely spaced examples are requested, and the results of F_c and $D_{(c,y_c)}$ are recalculated to greater accuracy. At the same time, this increases the precision of the coefficients of the matching pattern. The final result is a new function description, with more precise parameters.

Next, LINUS attempts to verify the description obtained by showing that the function it implies approximates the examples of the unknown function. Verification is performed both forwards and backwards. In forward verification, additional examples are requested that are evenly distributed within the range of the current ones, and the transformation sequence is performed on them to check that their transformed images match the quadratic pattern in the description. In backward verification, the inverse of the transformation sequence is applied to randomly-generated examples of the pattern and the results are compared with examples of the unknown function to check that the differences are less than the expected computation error.

5 Summary

Data transformation provides a foundation for function discovery systems that go well beyond the rational functions to which earlier schemes are restricted. This paper defines a data transformation language and characterizes its expressive power from a theoretical perspective. The

pioneering system FFD [6] had two fundamental weaknesses that seemed to be inherent in the data transformation approach: the potentially incomplete generation of function descriptions due to mathematical constraints on the transformations, and the inevitable errors caused by inexact computation of transformations. The present paper has indicated how both of these are overcome by LINUS. First, it is designed as an interactive system with adaptive example selection based on the on-going analysis of computation errors and transformations' requirements. Second, its transformations are more robust than those in FFD, and all can be applied with the appropriate selection of examples. Third, it is able to monitor the accuracy of its computations after after each transformation. Fourth, it includes a more flexible verification scheme to further improve the accuracy of all solutions found.

References

[1] B.C. Falkenhainer and R.S. Michalski. "Integrating quantitative and qualitative discovery: The ABACUS system." *Machine Learning*, **1**: 367–401, 1986.

[2] M.M. Kokar. "Determining arguments of invariant functional descriptions." *Machine Learning*, **1**: 403–22, 1986.

[3] P. Langley and J. Zytkow. "Data-driven approaches to empirical discovery." *Artificial Intelligence*, **40**, 283–312, 1989.

[4] P. Langley, J.M. Zytkow, H.A. Simon and G.L. Bradshaw. "The search for regularity: Four aspects of scientific discovery." In *Machine Learning: An Artificial Intelligence Approach, Volume II.* Morgan Kaufmann, 1986.

[5] B. Nordhausen and P. Langley. "A robust approach to numeric discovery." *Proc. International Conference on Machine Learning.* Morgan Kaufmann, 1990.

[6] P. Wong. *Machine Discovery of Function Forms.* PhD thesis, University of Waterloo, 1991.