7-24-2000

Piecewise Linear Homeomorphisms: The Scalar Case

Richard E. Groff
University of Michigan

Daniel E. Koditschek
University of Pennsylvania, kod@seas.upenn.edu

Pramod P. Khargonekar
University of Michigan

NOTE: At the time of publication, author Daniel Koditschek was affiliated with the University of Michigan. Currently, he is a faculty member in the Department of Electrical and Systems Engineering at the University of Pennsylvania.

Piecewise Linear Homeomorphisms: The Scalar Case

Abstract
The class of piecewise linear homeomorphisms (PLH) provides a convenient functional representation for many applications wherein an approximation to data is required that is invertible in closed form. In this paper we introduce the graph intersection (GI) algorithm for "learning" piecewise linear scalar functions in two settings: "approximation," where an "oracle" outputs accurate functional values in response to input queries; and "estimation," where only a fixed discrete data base of input-output pairs is available. We provide a local convergence result for the approximation version of the GI algorithm as well as a study of its numerical performance in the estimation setting. We conclude that PLH offers accuracy closed to that of a neural net while requiring, via our GI algorithm, far shorter training time and preserving desired invariant properties unlike any other presently popular basis family.

NOTE: At the time of publication, author Daniel Koditschek was affiliated with the University of Michigan. Currently, he is a faculty member in the Department of Electrical and Systems Engineering at the University of Pennsylvania.

Piecewise Linear Homeomorphisms: The Scalar Case*

Richard E. Groff Daniel E. Koditschek Pramod P. Khargonekar
EECS, University of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109 USA
regroff@eecs.umich.edu kod@eecs.umich.edu pramod@eecs.umich.edu

Abstract

The class of piecewise linear homeomorphisms (PLH) provides a convenient functional representation for many applications wherein an approximation to data is required that is invertible in closed form. In this paper we introduce the graph intersection (GI) algorithm for "learning" piecewise linear scalar functions in two settings we term "approximation" (where an "oracle" outputs accurate functional values in response to input queries) and "estimation" (where only a fixed discrete data base of input-output pairs is available). We provide a local convergence result for the approximation version of the GI algorithm as well as a study of its numerical performance (compared to truncated Taylor series approximation and to Neural Nets) in the estimation setting. We conclude that PLH i) offers nearly the accuracy of a Neural Net while ii) requiring, via our GI algorithm, the far shorter (several orders of magnitude less) training time typical of Taylor series approximants and iii) preserving desired invariant properties unlike any other presently popular basis family.

1 Introduction

The last decade has witnessed vigorous activity in the function approximation literature driven by a renewed interest in machine learning for engineering applications [1]. In our view, some of the most compelling questions in this field arise in the effort to determine the capabilities of nonlinear-in-parameters (NLIP) function families — e.g., the "neural nets" [2, 3] — relative to the classical properties of linear-in-parameters (LIP) bases [4]. Despite this significant level of activity, there remains a continuing controversy concerning the relative merits of one over another approximant in specific engineering settings. More recently, the class of "non-parametric" approximation methods has received increasing attention [5, 6] (from our point of view, it seems more natural to think of these as strongly NLIP representations) with the great advantage of widening the choice of approximant albeit threatening, perhaps, still more confusion as to the relative benefits of the various function families.

In this paper, we target the wide range of applications settings wherein it is known *a priori* that the function to be learned preserves certain invariants. As far as we are aware, the first mention of this idea in the machine learning literature occurs in a paper by Abu-Mostafa [7] wherein it is proposed to add "hints" describing *a priori* known features of the function being approximated, including, as a possible instance, specific invariance properties. Of course, there is the much longer tradition in the AI literature of what might be termed "invariant recall" that Abu-Mostafa attributes originally to Minsky and Papert [8]. In this paradigm, the problem of pattern recognition is cast as the computational effort to identify an archetype that presents itself in any particular instance as some "deformed" set of features. Namely, one assumes that the sensed data arises from the action upon the archetype of some unknown member of a known group of transformations. The pattern recognition task is to "factor out" the particularizing transform so as to reveal that "true" exemplar of the "concept." In contrast, as befits the learning paradigm, we are concerned here with the problem of identifying (via closest approximation) a specific member of the group of continuous and continuously invertible transformations of a Euclidean space, assuming from the start that the archetype — in this setting, the identity map — has already been specified.

Interest in these invariance properties was originally motivated by the approximation of color space transformations for color printing [9]. Many additional potential applications exist, such as robot navigation [10, 11], robot task description [12], machine tool calibration [13] and automobile fuel control settings [14], among others. More application specific motivation for this line of work is given in [15].

*This research was supported in part by an NSF GOALI, Award ECS-96322801

0-7695-0619-4/00 $10.00 © 2000 IEEE

We re-introduce the class of piecewise linear homeomorphisms (PLH) as a non-parametric (strongly NLIP) family of approximants that preserves the relevant invariant. [1] PLH has the advantage of being invertible in closed form, whereas most NLIP approximations can only be inverted numerically. We offer a numerical comparison in the context of scalar endomorphisms (invertible maps of a real interval into itself) comparing the performance of two commonly used approximants — the truncated Taylor series (TT), an LIP family; and the Neural Nets (NN), an NLIP family — with PLH. We conclude that PLH offers nearly the accuracy of NN while enjoying the invariance property that both TT and NN lack. We propose the Graph Intersection (GI) algorithm, a new fitting technique for PLH functions, and offer numerical evidence suggesting that it significantly accelerates the training time for PLH functions.

1.1 Background Literature

Piecewise linear approximations fall into a larger class of approximation schemes sometimes referred to as locally parametric (LP), in which the approximation is given over a partition of the domain. Adjusting a parameter changes the approximation over some limited number of partitions cells, i.e. there is a local, rather than global, change in the approximation. Lookup tables with an accompanying interpolation scheme are a simple form of nonparametric approximation. Locally parametric approximation leads naturally to nonparametric methods, where the quality of the approximation is improved by adjusting the number of cells in the partition, as well as the local parameters.

Atkeson et al. [16] use "locally linear experts," a locally parametric approximation scheme with a cover rather than a partition. The piecewise (possibly discontinuous) polynomial literature most directly affects the present work, though it works with the case where the underlying function is known in closed form, whereas we want to find continuous approximations by examining a limited data set. Barrow et al.[17] present generalized convexity conditions which guarantee the existence of a unique best approximant from the space of linear B-splines. Kioustelidis [18, 19] presents necessary conditions for an optimal piecewise polynomial approximation, as well as an algorithm for computing good piecewise polynomial approximations. Gayle and Wolfe [20] apply a modified version of the Kioustelidis algorithm and constructively re-establish the uniqueness results of Barrow, et al. under the same generalized convexity conditions, for piecewise polynomial approximations.

2 Graph Intersection Algorithm

2.1 Notation

Let $\mathcal{A}[c,d] \subset L_2[c,d]$ be the vector space of affine functions on the interval $[c,d] \subset \mathbb{R}$. Let $Q = \{Q_1, Q_2, ..., Q_n\}$ be a collection of n closed intervals, sequentially ordered, intersecting only at each successive neighbor's endpoints, whose union is the interval $\mathcal{I} = [a,b] \subset \mathbb{R}$. [2] In an abuse of notation, let $\mathcal{A}(Q)$ be the space of n cell piecewise affine, possibly discontinuous, functions. For $f \in \mathcal{A}(Q)$

$$f|_{Q_i} \in \mathcal{A}(Q_i)$$

Notice that for a given partition Q, $\mathcal{A}(Q)$ is a closed subspace of L_2. Let $\mathcal{CA}(Q) \subset \mathcal{A}(Q)$ be the linear subspace formed by all continuous n cell piecewise affine functions over Q.

The set (not a linear subspace) of n cell continuous piecewise affine functions over the domain $\mathcal{I} := [a,b] \subset \mathbb{R}$ can be parameterized by a set of $n+1$ points in the graph space[3] $\mathcal{G}$

$$P = \begin{bmatrix} q^{\mathrm{T}} \\ r^{\mathrm{T}} \end{bmatrix} \qquad q^{\mathrm{T}} = [q_0 \;\; q_1 \;\; \cdots \;\; q_n] \in \mathbb{R}^{n+1} \quad r^{\mathrm{T}} = [r_0 \;\; r_1 \;\; \cdots \;\; r_n] \in \mathbb{R}^{n+1} \tag{1}$$

with $a = q_0 < q_1 < \ldots < q_{n-1} < q_n = b$. ($q_0$ and q_n are fixed, so they are not parameters in the most narrow sense.) We call $p_i = (q_i, r_i)$ a vertex of the function and we refer to q as the domain partition vector, since q implies an n cell partition on the domain. We refer to r as simply the codomain vector, since thus far we do not require that it induce a partition on the codomain. We call the collection of

[1] Although this function class seems not to have been studied within the machine learning community, the usage "re-introduce" is intended to acknowledge the long and distinguished history of PLH in the spline literature and the more general world of pure mathematics.

[2] In the sequel, we will speak, colloquially, of such a set of subintervals as defining a partition of $\mathcal{I}$, even though it is actually the one dimensional simplicial complex of vertices and open subintervals that defines a formal partition.

[3] The graph space of a function $f : U \to V$ is $\mathcal{G} = U \times V$. The graph of f is the set $\{(x, f(x)) | x \in U\} \subset \mathcal{G}$.

domain partition cells $Q = \{Q_1, Q_2, \ldots, Q_n\}$, where $Q_i = [q_{i-1}, q_i]$. The line l_i which connects vertices p_{i-1} and p_i is given by,

$$l_i(x) \;=\; \alpha_i\,(x - q_{i-1}) + r_{i-1} \;=\; \alpha_i\,(x - q_i) + r_i \qquad \text{where } \alpha_i = \frac{r_i - r_{i-1}}{q_i - q_{i-1}} \tag{2}$$

The continuous piecewise function f_P defined by P is

$$f_P(x) = l_i(x) \quad \text{when } x \in Q_i$$

2.2 Algorithm

The graph intersection algorithm is motivated by a real world problem where a data set - a finite collection of input-output pairs - must be used to find a good n cell piecewise linear approximation to some hypothesized underlying function. We will refer to this setting as the estimation version of the G.I. algorithm.

Alternatively, the underlying function could be provided directly to the algorithm, rather than through data. We refer to this setting as the approximation version of the algorithm. Intuitively, the approximation version arises from the estimation setting supplied with an unbounded amount of data distributed uniformly across the domain. In [15] we provide a local convergence result for the approximation version of G.I. when the underlying function lies within the approximant's parameter space. i.e. the underlying function is also an n cell piecewise linear function.

We first state the Graph Intersection Algorithm on a conceptual level. The following sections will provide details for the estimation version along with numerical studies and for the approximation version with analtical results. The algorithm is initialized with P^0, the parameterization of any valid n cell piecewise linear function, as from Eqn. 1. (Iteration number will be denoted by a superscript, e.g. after the j^{th} iteration, the approximation is parameterized by P^j.)

Graph Intersection Algorithm

1. Partition data according to P^j
2. Compute the best affine map, m_i, in each cell Q_i^j.
3. For $i = 1, ..., n-1$,
   if m_i, m_{i+1} are not parallel, then let $g_i \in \mathcal{G}$ be their unique intersection point
   else $g_i = p_i^{(k)}$
4. For $i = 1, ..., n-1$,
   if g_i meets the Acceptance Condition, then $p_i^{j+1} = g_i$
   else choose p_i^{j+1} using a "Special Rule"
5. $j \leftarrow j + 1$. Goto step 1

In the estimation version, the GI algorithm is given a set of N input-output pairs, $\mathcal{Z} = \{d_j\}_{j=1}^N$. The data is sorted according to the partition, and the best linear map of Step 2 is the least squares linear fit to the data in the respective partition cell. In some circumstances, the best fit lines of neighboring cells have the same slope, precluding any intersection point (in the case of parallel lines) or yielding a continuum of intersection points (in the case of coincident lines). An equally important, but less obvious problem occurs when neighboring cells have best fit lines which are nearly parallel. This may cause the resulting intersection point to be far away from the previous vertex, potentially even outside of the domain of approximation, destroying the ordering of the partition vector q. In this case the new knot point must be picked in another manner, hence the Acceptance Condition and "Special Rules." For a detailed treatment of these conditions and rules see [15].

2.3 Numerical Results for the G.I. Estimation Problem

We studied the approximation of homeomorphisms of the interval $[0, 1]$ into itself from noiseless data using piecewise linear approximants (PL) as well as neural networks(NN) and taylor polynomials(TP) representing truncated Taylor series. These techniques are representative of LP, NLIP, and LIP approximations, respectively. Each approximation was given the same number of parameters. For example, the data shown here is for 16 parameters, corresponding to a NN with 5 neurons, a degree 15 TP and a PL approximant with 9 segments (the endpoints are fixed). All code was implemented in MATLAB. TP solves the standard linear least squares problem in closed form to find the coefficients. The NN has a

single hidden layer with hyperbolic tangent neurons and trains using Levenberg-Marquardt descent on the squared error. Results for two different forms of PL training are shown. The first (constr PL) uses MATLAB's constrained optimization algorithm directly on the squared error. The Graph Intersection (GIPL) algorithm is the second method for computing PL approximations.

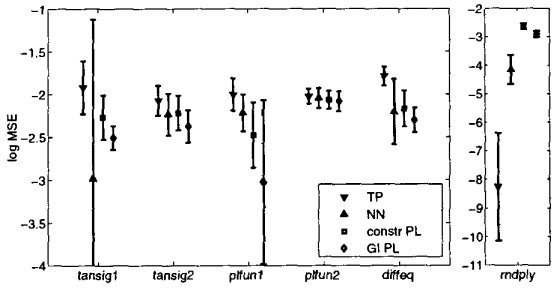

Figure 1: Mean and standard deviation bars for the Log Mean Squared Error on the six function families.

Table 1: Mean computational cost (in megaflops) of the approximation techniques on the six families.

	TP	NN	PL con	PL GI
`tansig1`	0.187	10.2	1.89	0.168
`tansig2`	0.187	9.38	1.68	0.240
`plfun1`	0.187	9.71	1.93	0.185
`plfun2`	0.187	9.74	2.12	0.538
`rndply`	0.187	10.7	1.02	0.157
`diffeq`	0.187	9.71	2.06	0.195

The choice of homeomorphisms, functions which are invertible, continuous, and have a continuous inverse, is motivated by the proposed applications, where a "change of coordinates" is learned between an *a priori* topological model with invariant properties and the world as represented by data. A change of coordinates is, of course, an invertible map. Six families of homeomorphisms were chosen for study. Families `tansig1` and `tansig2` are invertible superpositions of hyperbolic tangents, and families `plfun1` and `plfun2` are invertible piecewise linear. Families `tansig1` and `plfun1` lie within the representational power of the neural network and the piecewise linear approximant, respectively, whereas the approximations are "underparameterized" on `tansig2` and `plfun2`. The fifth family is high degree polynomials (`rndply`). These functions are generated by taking the composition of seven invertible quadratic functions. The last family is time one maps from differential equations (`diffeq`). Recall that the time one map of a differential equation with unique solutions is not only a homeomorphism but also a diffeomorphism. One hundred functions were randomly sampled from each of the six families.

Figure 1 summarizes the approximation error for the study. As expected, NN outperforms the other three techniques on the class `tansig1`, where the functions lie with the parameter space of the NN. Similarly both PL techniques outperform the other two techniques on `plfun1`, where the functions lie within the parameter space of PL. GI PL consistently produces better results than constr PL. On most families the PL techniques perform better than the other two techniques. An exception is the `rndply` family, on which NN did notably better than PL, but otherwise PL is competitive in terms of error, and also has the benefit of being closed form invertible.

Table 1 shows training cost in terms of floating point operations (flops) for the algorithms on the various function families. By far the highest computational cost belongs to NN, which is almost two orders of magnitude more intensive than TP. Note that constr PL is also computationally expensive. GI PL is close to TP in computational cost and is on some families actually uses less flops than TP, which is a linear-in-parameters approximation.

Overall, on these classes of homeomorphisms studied, GI PL is competitive in terms of squared error with both TP and NN. It is surprisingly computationally efficient. Moreover it is possible to easily check the invertiblity of the resulting function, as well as invert it in closed form.

2.4 Analytical Results for the G.I. Approximation Problem

In the approximation version, the Graph Intersection algorithm is supplied with a complete function in closed form, which is to be approximated by a piecewise linear function. In this case, the "best affine map" for a partition cell Q_i in step 2 of the algorithm is the best L_2 affine approximation of $\boldsymbol{f}$ over that cell. This is a linear projection, and may be thought of as the infinite data limit of the least squares map in the estimation version.

We have established a local convergence result for the approximation version of the graph intersection algorithm. Consider the case where the underlying function we wish to approximate, call it $f_P^* : \mathcal{I} \to \mathbb{R}$, lies within the parameter space of the approximant. That is, $f_P^* \in \mathcal{CA}(Q^*)$, where Q^* is an n cell partition of the interval $\mathcal{I}$. Thus the function f_P^* may be parameterized as described in §2.1. Parameters of the underlying function f_P^* will be denoted by a superscript * in order to distinguish them from the parameters of the approximation.

Given an underlying piecewise affine function f_P^*, if the domain values of the vertices of the approximation f_P start close enough to the domain values of the vertices of the underlying function f_P^*, then application of the graph intersection algorithm causes the approximation to converge to the underlying function. Stated more formally:

Theorem 1 *Let $f_P^* \in \mathcal{CA}(Q^*)$, where Q^* is an n cell partition of $\mathcal{I}$. There exists $\epsilon_0 > 0$ such that if the G.I. algorithm is initialized with an n cell approximant such that $\left\|q^0 - q^*\right\|_\infty < \epsilon_0$, then iteration of the algorithm satisfies*

$$\lim_{j \to \infty} \left\| f_P^j - f_P^* \right\|_\infty = 0.$$

A detailed treatment of the approximation version of the G.I. algorithm with a proof of the theorem is in [15].

3 Conclusion

The use of piecewise linear approximations is motivated by applications wherein certain invariants, captured by model functions between model spaces are to be preserved by a "learned" change of coordinates — a piecewise linear homeomorphism. A scalar piecewise linear function is trivially in PLH when its defining domain and codomain subintervals are arranged in an "ordered" partition, in which case it's closed form inverse function is computed by simply interchanging the domain and codomain partitions. Missing, heretofore, from the view of engineering applications, has been an efficient algorithm for computing PL approximations from finite sampled data sets.

This paper has introduced a PL function training method that compares favorably respecting both accuracy and speed with popular linear- and nonlinear-in-parameters function approximation techniques, as summarized in Figure 1 and Table 1 of Section 3. The Graph Intersection Algorithm exploits the structure of piecewise linear maps to provide a training method that approaches (and apparently, sometimes surpasses) the computational simplicity of linear in parameters representations (TP in Figure 1) yet retains the approximation power of PL trained by traditional constrained hill climbing methods (constr PL in Figure 1) which, in turn, approaches (and, apparently, sometimes surpasses) the neural nets (NN in Figure 1) in approximation accuracy. The paper concludes with a local convergence proof for the approximation version of the new algorithm.

While the present work has focused on scalar maps, it is clear that many interesting applications settings present the need for a multidimensional extension of these ideas. We are currently exploring the performance of a version of GI generalized to higher dimensions [9]. Preliminary indications are sufficiently favorable that a more concerted effort to understand the convergence properties of GI style algorithms would now seem to be in order. For example, the proof offered here presumes that the underlying function to be learned is piecewise linear with the same number of cells — evidently not a requirement for good performance. Similarly, the effect of varied "Special Rules" and the best choice of high dimensional generalization remain to be illuminated by analysis.

Acknowledgements: Conversations with Roger Brockett, Rob Ghrist, Moe Hirsch, and Dave Neuhoff have substantially refined the technical vision underlying this work. Tracy Thieret and LK Mestha have provided invaluable guidance concerning its potential industrial relevance. We thank Stefan Schaal for a number of stimulating discussions and Elmer Gilbert for several useful suggestions regarding the presentation here.

References

[1] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[3] Andrew R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[4] Philip J. Davis, *Interpolation and Approximation*, Dover Publications, Inc., 1975.

[5] Jerome H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[6] Charles J. Stone, Mark H. Hansen, Charles Kooperberg, and Young K Truong, "Polynomial splines and their tensor products in extended linear modeling," *The Annals of Statistics*, vol. 25, no. 4, pp. 1371–1470, 1997.

[7] Yaser S.Abu-Mostafa, "Learning from hints," *Journal of Complexity*, vol. 10, pp. 165–178, 1994.

[8] Marvin Minsky and Seymour Papert, *Perceptrons : an introduction to computational geometry*, MIT Press, 1969.

[9] Richard Groff, Pramod Khargonekar, Daniel Koditschek, Tracy Thieret, and L.K. Mestha, "Modeling and control of xerographic processes," in *Proceedings of the 38th IEEE Conference on Decision and Control*, 1999.

[10] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Transactions of the American Mathematical Society*, vol. 327, no. 1, pp. 71–115, Sep 1991.

[11] Elon Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct 1992.

[12] R. W. Brockett, "On the computer control of movement," in *Proc. IEEE Conference on Robotics and Automation*, pp. 534–540. IEEE, Philadelphia, PA., 1988.

[13] Neiyuan Hai, Jun Ni, and Jingxia Yuan, "Generalized model formulation technique for error synthesis and error compensation on machine tools," in *Proceedings of the NAMRX XXVI Conference.* Society of Manufacturing Engineers, 1998.

[14] Scott Furry and Jeff Kainz, "Rapid algorithm development applied to engine management systems," in *Proc. 1998 SAE Intl Congress & Exposition.*

[15] R.E. Groff, P.P. Khargonekar, and D.E. Koditschek, "Training piecewise linear homeomorphisms for approximation of functions with known invariants: the scalar case," UM CSE Tech Report.

[16] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1–5, pp. 11–73, Feb 1997.

[17] D. L. Barrow, C. K. Chui, P. W. Smith, and J. D. Ward, "Unicity of best mean approximation by second order splines with variable knots," *Mathematics of Computation*, vol. 32, no. 144, pp. 1131–1143, October 1978.

[18] J. B. Kioustelidis, "Optimal segmented approximations," *Computing*, vol. 24, no. 1, pp. 1–8, 1980.

[19] J. B. Kioustelidis, "Optimal segmented polynomial L_s-approximations," *Computing*, vol. 26, no. 3, pp. 239–246, 1981.

[20] R. C. Gayle and J. M. Wolfe, "Unicity in piecewise polynomial L_1-approximation via an algorithm," *Mathematics of Computation*, vol. 65, no. 214, pp. 647–660, April 1996.