Improving the Approximation and Convergence Capabilities of Projection Pursuit Learning

March 14, 1995

Abstract

One nonparametric regression technique that has been successfully applied to high-dimensional data is projection pursuit regression (PPR). In this method, the regression surface is approximated by a sum of empirically determined univariate functions of linear combinations of the predictors. Projection pursuit learning (PPL) proposed by Hwang *et al.* formulates PPR using a two-layer feedforward neural network. One of the main differences between PPR and PPL is that the smoothers in PPR are nonparametric, whereas those in PPL are based on Hermite functions of some predefined highest order R. While the convergence property of PPR is already known, that for PPL has not been thoroughly studied. In this paper, we demonstrate that PPL networks do not have the universal approximation and strong convergence properties for any finite R. But, by including a bias term in each linear combination of the predictor variables, PPL networks can regain these capabilities, independent of the exact choice of R. It is also shown experimentally that this modification improves the generalization performance, and creates smoother decision surfaces for classification problems.

1 Introduction

In recent years, many neural network models have been proposed for pattern classification, function approximation and regression problems. Among them, the class of multi-layer feedforward networks is perhaps the most popular. Standard back-propagation performs gradient descent in the weight space of a network with fixed topology, and is useful only when the network architecture is chosen correctly. Too small a network cannot learn the problem well, but a size too large will lead to over-generalization and thus poor performance. Hence, some recent studies have sought to take the *constructive* approach [1, 2], which starts with a small network and then grows additional hidden units and weights until a satisfactory solution is found.

Besides learning the weights and network size, one may also modify the transfer functions in the hidden units, because different transfer functions are suitable for different problems. By making their functional forms modifiable by a set of parameters, the transfer functions can adapt themselves under different situations. Moody [3], for example, considered a number of these flexible transfer functions and demonstrated experimentally better generalization performance as compared to the sigmoid function.

2 PPL as Generalization of PPR

Projection pursuit learning (PPL)[1] [5, 6] is a constructive procedure developed along the lines as discussed above. It is inspired from the statistical technique projection pursuit regression (PPR) [7]. In a regression problem, one is given a d-dimensional random vector $\mathbf{X}$, the components of which are called predictor variables, and a random variable, Y, called response. A regression surface f describes a general relationship between $\mathbf{X}$ and Y. Without loss of generality, we assume $E(f) = 0$. In PPR, f is approximated by:

$$f_{n,PPR}(\mathbf{x}) = \sum_{j=1}^{n} g_j(\mathbf{a}_j^T \mathbf{x}), \tag{1}$$

where $\mathbf{a}_j^T \mathbf{x}$ denotes the inner product of the projection vector $\mathbf{a}_j$ (with $||\mathbf{a}_j|| = 1$) and input vector $\mathbf{x}$, and g_j is called the *smoother*. The main advantage of PPR is that it does not suffer from the "curse of dimensionality".

PPL is obtained by formulating PPR using a two-layer feedforward neural network. Without loss of generality, we consider networks with only one output unit. The output $f_n(\mathbf{x})$ for a network with n hidden units is:

$$f_n(\mathbf{x}) = \sum_{j=1}^{n} \beta_j g_j(\mathbf{a}_j^T \mathbf{x}), \tag{2}$$

where β_j is the weight connecting the jth hidden unit to the output unit, g_j is the transfer function for the jth hidden unit, and the components of $\mathbf{a}_j$ are the weights connecting all the input units to the jth hidden unit with $||\mathbf{a}_j|| = 1$. The similarity between (1) and (2) should be apparent.

One of the main differences between PPR and PPL is in the form of the smoothers. In PPR, the smoothers are nonparametric and are usually based on locally linear fits [7]. However, this leads to the use of large regression tables, unstable approximation in calculating derivatives, and piecewise interpolation [5].

[1]This term was also coined by [4]. But, in this paper, we refer PPL to the formulation by Hwang *et al.* [5].

On the other hand, the smoothers in PPL are parametric. They are represented as linear combinations of Hermite functions[2] [8] of the form

$$g_j(\mathbf{a}_j^T\mathbf{x}) = \sum_{r=1}^{R} c_{jr} h_{jr}(\mathbf{a}_j^T\mathbf{x}), \tag{3}$$

where R, called the *order*, is a constant set by the user. The h_r's are orthonormal and defined by $h_r(z) = \frac{1}{\sqrt{2\pi}}(r!)^{-1/2}\pi^{1/4}2^{-(r-1)/2}H_r(z)e^{-z^2/2}$, where $H_r(z)$'s are the Hermite polynomials. The use of Hermite functions enables smooth interpolation and also fast and accurate computation of the derivatives without using large regression tables. Moreover, the optimal coefficients for β_j's and c_j's may be computed by linear algebra, so the only part that requires nonlinear optimization techniques is the $\mathbf{a}_j$'s. Experimentally, PPL networks are considerably more parsimonious and accurate than conventional back-propagation networks on a number of regression problems [5]. For classification problems, PPL networks are also able to produce smoother classification boundaries than the cascade-correlation architecture, and are thus expected to generalize better [6].

However, a major problem with PPL is that the order R has to be chosen correctly for good training and testing performance. This is usually explained as a manifestation of the bias-variance dilemma. By using a large R, one is supposedly able to decrease the bias while possibly increasing the variance. In the following section, we will demonstrate that besides the above reason, another important reason is that with a fixed R, the bias cannot be made as small as desired by increasing n even without bound. In other words, PPL networks with a fixed R do not have the property of universal approximation.

3 Fixing R Hampers Universal Approximation

The strong convergence of PPR has been proved in [9], which states that if each new g_n in (1) at stage n is given by $g_n(z) = E(f - f_{n-1}|\mathbf{a}_n^T\mathbf{X} = z)$, and the projection direction $\mathbf{a}_n$ is chosen such that $E(g_n(\mathbf{a}_n^T\mathbf{X}))^2 > \rho \sup_{\mathbf{b}^T\mathbf{b}=1} E(g_n(\mathbf{b}^T\mathbf{X}))^2$, where $0 < \rho < 1$ is fixed, then f_n in (1) strongly converges to the desired f. However, this result is not readily applicable to PPL, as has been assumed in [5]. With the smoothers in PPL being *parametric*, this g_n may not always be realizable. It is this very concern that has led us to the research work reported here.

As a simple illustration, consider the case when $d = 1$. Put $z = ax$. The polynomials $h_r(z)$ are even or odd functions according to whether the index r is even or odd [8]. Besides, as $|a|$ is restricted to be 1, there can be at most two projection directions for the hidden units,

[2] Other parametric forms may be used in place of the Hermite functions, but the pros and cons among these varieties will not be addressed in this paper.

corresponding to $z = \pm x$. Assume that there are n_1 hidden units with $z = x$, and n_2 with $z = -x$, where $n_1 + n_2 = n$. From (2), the network output f_n is given by:

$$\begin{aligned} f_n(x) &= \sum_{j=1}^{n_1} \alpha_j g_j(x) + \sum_{j=1}^{n_2} \beta_j g_j(-x) \\ &= \sum_{j=1}^{n_1} \sum_{r=1}^{R} \alpha_j c_{jr} h_r(x) + \sum_{j=1}^{n_2} \sum_{r=1}^{R} \beta_j d_{jr} h_r(-x) \\ &= \sum_{s=1}^{\lfloor R/2 \rfloor} (p_{2s} + q_{2s}) h_{2s}(x) + \sum_{s=1}^{\lfloor (R-1)/2 \rfloor} (p_{2s+1} - q_{2s+1}) h_{2s+1}(x), \end{aligned}$$

where $p_r = \sum_{j=1}^{n_1} \alpha_j c_{jr}$, $q_r = \sum_{j=1}^{n_2} \beta_j d_{jr}$. Since the family of Hermite functions is complete[3] in $L^2(-\infty, +\infty)$ only when R is infinite [8], the universal approximation property of PPL networks does not hold for any finite R. As a consequence, in general, the sequence $\{f_n\}$ produced by the PPL procedure may not converge to the desired function f. This fact will also be experimentally demonstrated in Section 5.1.

4 Modified PPL with Addition of Bias Term

To remedy the problem suggested above, one has to determine the value of R so that it is "sufficiently" large for the problem at hand. One possibility could be to set R to be very large. However, a large R implies a large number of parameters, which may degrade generalization. Moreover, the number of computational steps, both during training and testing, is increased. Besides, a large R also increases the number of "flat spots" [10], which are locations where the derivative of the hidden unit transfer function approaches zero. This increases the chance that the hidden units will get stuck, making the optimization problem more difficult.

A more disciplined approach is to perform PPL at several fixed values of R, and compare the resultant networks using criteria such as AIC [11]. Note that because both R and the number n of hidden units affect the approximation accuracy, one has to make comparisons across different combinations of R and n, which is also very computationally expensive.

In the following, we suggest that one can keep R fixed, while still capable of achieving universal approximation simply by including a bias term into each linear combination of the predictors in (2), i.e.

$$f_n(\mathbf{x}) = \sum_{j=1}^{n} \beta_j g_j(\mathbf{a}_j^T \mathbf{x} + \theta_j).$$

[3]Note that this is required for universal approximation, not to mention exact representation.

4.1 Universal Approximation

The set of functions implementable by a modified PPL network with n hidden units is $S_d^n(\psi) = \{f_n : \Re^d \rightarrow \Re \mid f_n(\mathbf{x}) = \sum_{j=1}^n \beta_j \psi(\mathbf{a}_j^T \mathbf{x} + \theta_j)\}$. Consider $S_d(\psi) = \bigcup_{n=1}^\infty S_d^n(\psi)$. For $\psi(z) = ze^{-z^2/2}$, ψ is obviously bounded and nonpolynomial. Hence, by Theorem 2 of [12], $S_d(\psi)$ is dense in $L^p(\mu)$ for all compactly supported finite measures μ on $\Re^d$ and $1 \leq p < \infty$.[4] As the functional form of g in (3) subsumes that of ψ, hence with $z = \mathbf{a}^T\mathbf{x} + \theta$, neural networks with one layer of hidden units of the form (3) are universal approximators. In other words, the modified PPL networks are capable of universal approximation. In comparison with PPR [7], although the bias is not used, this is not a problem as the smoothers are nonparametric. It is also obvious that the order R in (3) does not affect the universal approximation property. In fact, any function containing $e^{-z^2/2}$ can be used as g in (3).

4.2 Strong Convergence

PPL constructs the network by adding hidden units one at a time. So when a new hidden unit is added,

$$f_n(\mathbf{x}) = \sum_{j=1}^{n-1} \tilde{\beta}_j \tilde{g}_j(\tilde{\mathbf{a}}_j^T \mathbf{x} + \tilde{\theta}_j) + \tilde{\beta}_n g_n(\mathbf{a}_n^T \mathbf{x} + \theta_n),$$

where $\tilde{\beta}_j, \tilde{g}_j, \tilde{\mathbf{a}}_j, \tilde{\theta}_j$ are the updated values of $\beta_j, g_j, \mathbf{a}_j$ and θ_j, respectively. Details on how to do the update are described in [5]. PPL can be implemented with or without *backfitting* [7, 5], which consists of cyclically adjusting the parameters associated with each previously installed hidden unit by minimizing the residual error until there is no significant change. Obviously, if backfitting is employed, all the weights are freely modifiable and thus strong convergence of the PPL procedure follows readily from the universal approximation capability of the modified PPL network. If backfitting is not performed, i.e. $\tilde{g}_j = g_j, \tilde{\mathbf{a}}_j = \mathbf{a}_j$ and $\tilde{\theta}_j = \theta_j$ for $j = 1, 2, \ldots, n-1$, it then follows from [14] and the universal approximation capability of the modified PPL networks that the sequence $\{f_n\}$ still strongly converges to f in L^2 provided that at each iteration when a new hidden unit is added, the $\tilde{\beta}_j$'s, $g_n, \mathbf{a}_n$ and θ_n are chosen so as to minimize the expression $||f - f_n||_2$.

[4] If we further drop the restriction of $||\mathbf{a}_j|| = 1$ in (2), then by Theorem 1 of [13], $S_d(\psi)$ is dense in $L^p(\mu)$ for all finite but not necessarily compactly supported measures μ on $\Re^d$.

5 Simulation Experiments

The implementation of our modified PPL algorithm is based on the C code of the original PPL algorithm provided by Jeng-Neng Hwang and Shyh-Rong Lay. But in the simulation below, we do not use the backward pruning procedure mentioned in [5] in order to focus on the issue of approximation capability.

5.1 Inadequacy of a Fixed R

Consider approximating the Hermite function $h_3(x)$:

$$f(x) = h_3(x) = \frac{2x^3 - 3x}{\sqrt{3}\pi^{1/4}} \exp(\frac{-x^2}{2}),$$

using $R = 2$.[5] A training set of 1000 points is randomly generated from the uniform distribution $U[-5, 5]$, and a test set of 2000 points is generated with regularly spaced intervals on $[-5, 5]$. Mean squared error ($MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - f_n(x_i))^2$, where N is the number of data points) is used for comparison.

The training and testing curves are shown in Figure 1. The original PPL algorithm cannot learn the function, while the use of a bias term in the modified algorithm allows the Hermite functions to shift for more accurate approximation.

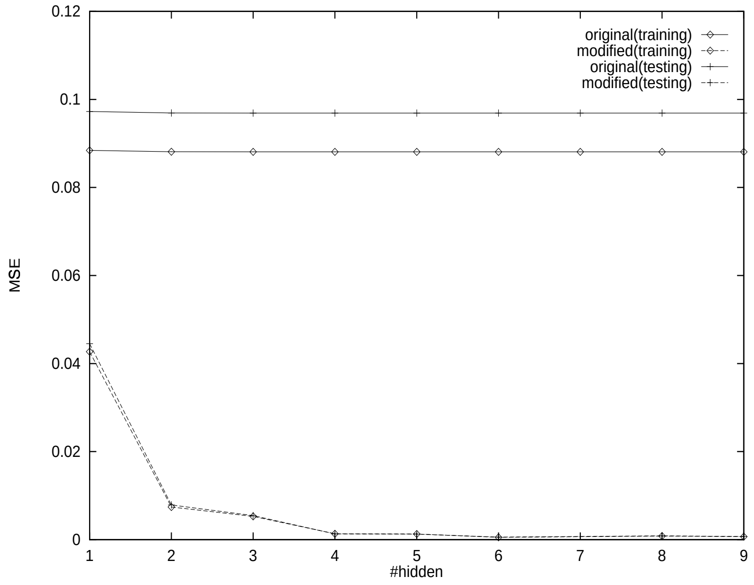

Figure 1: Training and testing curves for $f(x) = h_3(x)$ using $R = 2$.

[5]Obviously, using $R > 2$ would make exact representation trivial.

5.2 Regression Problems

The regression functions tested here are the complicated interaction function described in [5]:

$$f^{(1)}(x_1, x_2) = 1.9(1.35 + e^{x_1} \sin(13(x_1 - 0.6)^2) e^{-x_2} \sin(7x_2)),$$

and the 6-dimensional problem used in [15]:

$$f^{(2)}(x_1, x_2, x_3, x_4, x_5, x_6) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5.$$

Note that x_6 in $f^{(2)}$ is a spurious predictor that does not influence the response. Methods for generating the data are described in [15, 5]. Reported below is the *fraction of variance unexplained* ($FVU = \sum_{i=1}^{N}(f(\mathbf{x}_i) - f_n(\mathbf{x}_i))^2 / \sum_{i=1}^{N}(f(\mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i))^2$). The testing (i.e. generalization) performance for $R = 9$ is summarized in Table 1. With the modified algorithm, generalization can be improved.

Table 1: Comparison of testing (generalization) FVU with $R = 9$.

	noiseless $f^{(1)}$ (3 hidden)	noisy $f^{(1)}$ (3 hidden)	noiseless $f^{(1)}$ (5 hidden)	noisy $f^{(1)}$ (5 hidden)	$f^{(2)}$ (3 hidden)
original	0.05555	0.06399	0.04973	0.05653	0.499337
modified	0.05528	0.06395	0.02018	0.05627	0.266393

5.3 Classification Problem

The benchmark chosen is the two-spirals problem [1, 6]. The classification boundaries learned are shown in Figure 2. The modified PPL is able to generate a smoother classification boundary than the original one.

6 Conclusion

In this paper, we studied the fundamental issues of universal approximation and convergence capabilities of the PPL algorithm. We demonstrated that PPL networks do not have these capabilities for any finite order R. This helps to explain why the highest order R of the Hermite functions used in the PPL smoothers has a critical effect on the network's approximation accuracy. Moreover, note that while both R and the global bandwidth parameter in nonparametric smoothers are responsible for controlling the bias-variance tradeoff, they are not totally

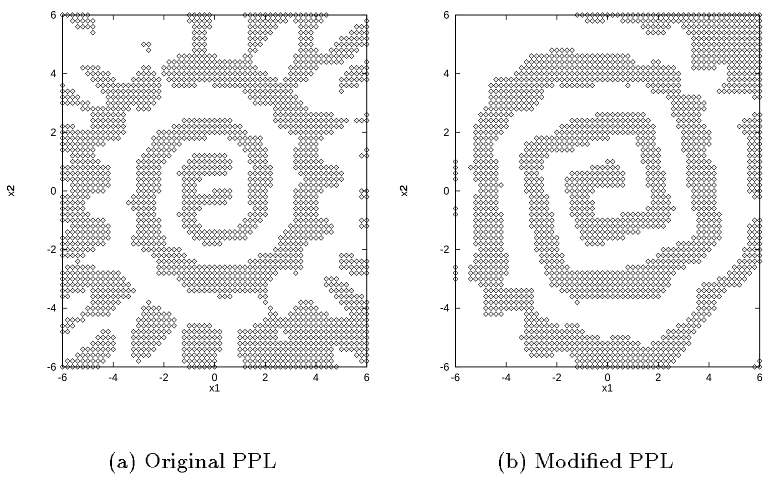

Figure 2: Classification boundaries for the two-spirals problem with 11 hidden units and $R = 13$.

equivalent, as suggested in [5]. Consider, for example, radial basis function networks in which the bandwidth corresponds to the "width" of each kernel (or hidden unit). As shown in [16], one can have just a global bandwidth (i.e., with the same width for all kernels) while still ensuring universal approximation, under certain regularity conditions on the functional form of the kernels. More relevant to our study here, this approximation capability is independent of the value of bandwidth chosen. This is however not the case for R in the absence of a bias term, because then the universal approximation property does not hold for any fixed finite R and thus the approximation error (i.e. bias) cannot be made as small as desired by trading variance.

By including a bias term in each linear combination of the predictor variables, PPL networks can regain both the universal approximation and convergence capabilities. Besides, this is not affected by the exact choice of R. We also showed experimentally that this modification improves the testing performance in regression problems and leads to smoother decision boundaries in classification problems.

In our future work, we will compare the use of Hermite functions as parametric smoothers with other possibilities. We will also compare PPL networks and conventional back-propagation networks with respect to the effective number of parameters.

7 Acknowledgments

The authors would like to thank Jeng-Neng Hwang and Shyh-Rong Lay for providing the C code of their PPL algorithm.

References

[1] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufmann, Los Altos CA, 1990.

[2] D.Y. Yeung. Constructive neural networks as estimators of Bayesian discriminant functions. *Pattern Recognition*, 26(1):189–204, 1993.

[3] J. Moody and N. Yarvin. Networks with learned unit response functions. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 1048–1055. Morgan Kaufmann, 1992.

[4] Y. Zhao and C.G. Atkeson. Projection pursuit learning. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 869–874, Seattle, WA, USA, July 1991.

[5] J.N. Hwang, S.R. Lay, M. Maechler, D. Martin, and J. Schimert. Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on Neural Networks*, 5(3):342–353, May 1994.

[6] J.N. Hwang, S.S. You, S.R. Lay, and I.C. Jou. What's wrong with a cascaded correlation learning network: A projection pursuit learning perspective. Submitted to *IEEE Transactions on Neural Networks.*

[7] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981.

[8] G. Sansone. *Orthogonal Functions.* Dover, New York, 1991.

[9] L.K. Jones. On a conjecture of Huber concerning the convergence of projection pursuit regression. *The Annals of Statistics*, 15(2):880–882, 1987.

[10] S.E. Fahlman. Faster learning variations on back-propagation: An empirical study. In D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, Los Altos, CA, 1988. Morgan Kaufmann.

[11] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723, December 1974.

[12] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6:1069–1072, 1993.

[13] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[14] L.K. Jones. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics*, 20(1):608–613, 1992.

[15] J.H. Friedman, E. Grosse, and W. Stuetzle. Multidimensional additive spline approximation. *SIAM Journal of Scientific and Statistical Computing*, 4(2):291–301, June 1983.

[16] J. Park and I. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.