ADAPTIVE LOGIC NETWORKS

WILLIAM W. ARMSTRONG, KENNETH O. COGGER

Kenneth O. Cogger, my longtime friend and collaborator, died on August 15, 2015. His contributions to statistical questions concerning ALNs have been extremely valuable. It is fitting that this paper and the ALNfitDeep software which is based on it be dedicated to Ken.

Abstract. An adaptive logic network (ALN) of dimension n computes either a real-valued, inhomogeneous linear function with $n-1$ real arguments, or a tree composition of such functions having two-input minimum and maximum operators at the non-leaf nodes. Smoothed ALNs which are once- or twice-continuously differentiable are obtained by inserting quadratic or quartic fillets into the node operators. Given samples of a continuous real-valued function f with random noise added, approximation of f by an ALN involves changing the coefficients of its linear pieces and growing the tree by splitting some leaf nodes into two. ALNs can fit any continuous function to any accuracy uniformly on a compact domain. The global noise level in data can be estimated by comparing output values on a set of samples, to the values interpolated by an ALN over-trained on a different set of samples. Subsequently, overtraining of smoothed ALNs is reduced by not splitting a piece when the root-mean-square error on the training data is less than the noise level estimate. Smoothed ALNs can be trained so that predetermined bounds on the first partial derivatives hold everywhere, including cases where monotonicity of the approximant with respect to any given input is assured. Given an ALN which is strongly monotonic in some variable, the inverse function can be computed by a trivial modification of the ALN.

Date: 2016.08.15.

1991 *Mathematics Subject Classification.* scientific.

Key words and phrases. machine learning, piecewise affine, regression, non-parametric, non-linear.

Contents

1. Definition and evaluation of an adaptive logic network (ALN)

Let $n \geq 2$. An ALN of dimension n is a labeled binary tree defined recursively as follows:

(1) A node labeled by a real-valued affine (inhomogeneous linear) function of $n-1$ real arguments is an ALN of dimension n.
(2) Two ALNs A_1, A_2 of dimension n which are the children of a node labeled either by minimum $\wedge$ (or resp. maximum $\vee$) form an ALN of dimension n denoted by $\wedge(A_1, A_2)$ (or resp. $\vee(A_1, A_2)$.
(3) Only binary trees constructed by a finite number of steps (1) and (2) above are ALNs of dimension n.

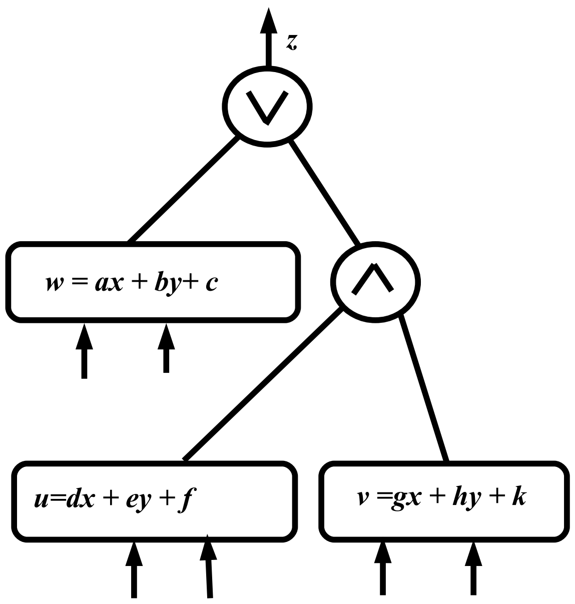


FIGURE 1. A simple ALN of dimension 3

The expression computed by the ALN in Fig. 1 is $z = \vee(w, (\wedge(u, v))$. A simple case which has qualitatively the same form is $z = \vee(x + y - 10, \wedge(x, y))$. The graph of that function shown is suggested in Fig. 2.

Usually the dimension n, which is the dimension of the graph of the ALN function and the number of coefficients in each linear function, will not need to be made explicit. When the arguments are numerical, we let their minimum (or resp. maximum) be denoted with the operator $\wedge$ (or resp.$\vee$)

If A is an ALN of dimension n, and $\mathbf{x} \in \mathbf{R}$, then the value of A at $\mathbf{x}$, denoted by $A(\mathbf{x})$, is defined as follows:

(1) If A is a node labeled by f, then $A(\mathbf{x}) = f(\mathbf{x})$.
(2) If $A = \wedge(A_1, A_2)$ then $A(\mathbf{x}) = \wedge(A_1(\mathbf{x}), A_2(\mathbf{x}))$,
(3) If $A = \vee(A_1, A_2)$ then $A(\mathbf{x}) = \vee(A_1(\mathbf{x}), A_2(\mathbf{x}))$,

The function computed by an ALN is piecewise linear and continuous (PWLC). ALNs have a universal approximation property as follows. Let D be a compact set of $\mathbf{R}^{n-1}$, and let $f : D \to \mathbf{R}$ be a continuous, real-valued function on D. Then, given any $\epsilon > 0$, there is an ALN that approximates f to within ϵ uniformly on D. There is a proof in Appendix A.

Lazy evaluation of ALNs is possible, for example if $A = \vee(A_1, \wedge(A_2, A_3))$, $A_1(\mathbf{x}) = 10$ and $A_2(\mathbf{x}) = 9$, then there is no point to evaluating $A_3(\mathbf{x})$. One of the

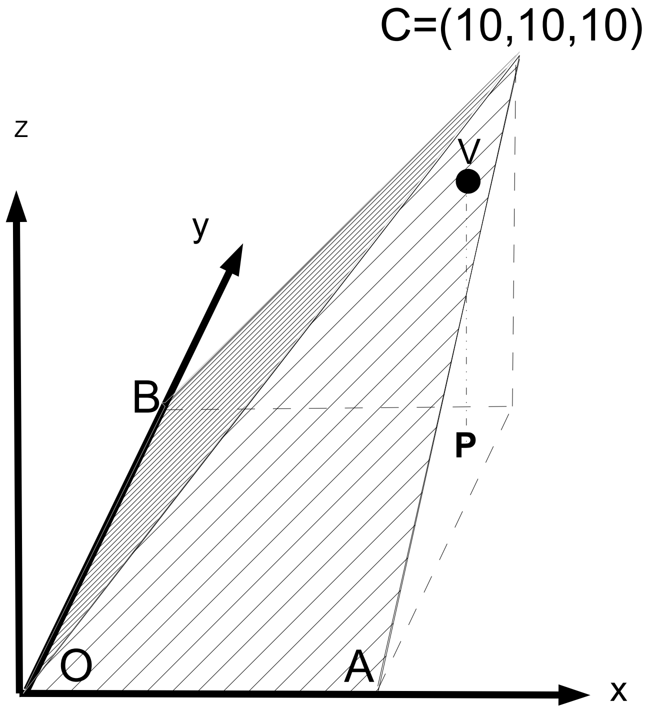


FIGURE 2. Part of the graph in three-dimensional space of a simple ALN. The plane (not shown) through points ABC is penetrated by a wedge formed by half-planes through AOC and BOC. Notice how the $\vee$ and $\wedge$ form a rudimentary saddle-surface. An appropriate ALN-fit to a curve, surface etc. should suggest the curvatures in the latter.

significant advantages of ALNs is that it is almost always possible to systematically omit a significant fraction of the evaluation steps above.

2. Relationship between data samples and pieces

Let A be an ALN of dimension n, and let $\mathbf{x} \in \mathbf{R}^{n-1}$. To find the set of linear pieces to which $\mathbf{x}$ belongs, do the following recursively.

(1) If A consists of a single linear piece, then $\mathbf{x}$ belongs to the the singleton set containing just that linear piece.

(2) If $A = \vee(A_1, A2)$ or $A = \vee(A_2, A_1)$ and $A_1(\mathbf{x}) > A_2(\mathbf{x})$ then $\mathbf{x}$ belongs to the same set of linear pieces as it does in A_1.
(3) If $A = \wedge(A_1, A2)$ or $A = \wedge(A_2, A_1)$ and $A_1(\mathbf{x}) < A_2(\mathbf{x})$ then $\mathbf{x}$ belongs to the same set of linear pieces as it does in A_1.
(4) If in (2) and (3) $A_1(\mathbf{x}) = A_2(\mathbf{x})$ then $\mathbf{x}$ belongs to the union of the sets it belongs to in A_1 and A_2.

In fig. 2, the point P belongs to the piece through AOC, and the value of the ALN at P is the z-value of the point V directly above it.

We note that for a randomly chosen $\mathbf{x}$, it is unlikely that case (4) will arise. Usually $\mathbf{x}$ will belong to a single linear piece of A. This observation leads us to an important feature of ALN learning. In learning systems which involve changing the values of a large number of numerical parameters to best approximate the data, finding a small set of parameters to change in order to reduce the error on given sample is called the "credit assignment problem". ALNs offer a very simple solution to that problem. Readers familiar with multilayer perceptron learning using the backpropagation algorithm will appreciate the elegance and simplicity of the ALN approach.

ALN credit assignment rule: To improve the approximation $A(\mathbf{x})$ to a specified output value y at input $\mathbf{x}$, change the weights on the piece or pieces of A to which $\mathbf{x}$ belongs.

The concept of an input $\mathbf{x}$ belonging to a piece will be generalized for smooth ALNs: credit must then be shared smoothly among several neighboring pieces.

3. Representation of inhomogeneous linear functions

We shall represent vectors (shown in boldface) as single-column matrices. A T superscript indicates transposition. Inhomogeneous linear functions f on D can be represented by a weight vector

$$\mathbf{w} = (w_1, ..w_n)^T \in \mathbb{R}^n$$

and a "centroid" vector

$$\mathbf{c} = (c_1, \ldots, c_n)^T \in \mathbb{R}^n.$$

Consider a point $\mathbf{x} = (x_1, \ldots, x_n)^T$. Then for all $(x_1, \ldots, x_{n-1})^T \in D$ with $f(x_1, \ldots, x_{n-1}) = x_n$, x_n is the unique solution of $(\mathbf{x} - \mathbf{c})^T \cdot \mathbf{w} = 0$. For this to work, the weight on the output variable must be non-zero. The weight vector $\mathbf{w}$ can be multiplied by any nonzero scalar without changing f. By convention we set the weight on the output variable to be -1. In this way, for example, $w_1(x_1 - c_1) + w_2(x_2 - c_2) = 0$ turns into $f(x_1) = x_2 = w_1(x_1 - c_1) + c_2$.

One reason for introducing a "centroid" $\mathbf{c}$ which, we shall see, should be situated close to to the samples $\mathbf{x}$ belonging to a linear piece, both in the input and output axes, is to try to avoid the loss of numerical accuracy which can happen when two large-magnitude floating point numbers are added or subtracted to give a small-magnitude result.

4. Determining the level of noise in the data

Given a set of samples (X_k, y_k) of inputs and the corresponding outputs, if no two input vectors are the same, then one could conclude that the data points all lie on the function one is looking for and there is no noise. We usually know this is not the case. On the other hand, if one has a large number of samples showing different output values for identical inputs, then one could estimate the noise in the data by computing the variance v of the set of outputs for a given input. Usually though, there will be few, if any, pairs of samples with identical inputs. To overcome this problem, ALNfit Pro divides the set of samples into two parts, one of which we'll call the training set and the other the validation set. It can use the training set and its y-values to interpolate y-values for any input vector not in the training set. In particular it can interpolate a y-value for an input in the validation set. This gives us a pair of output values for the same input vector. Now if we know the variance v' of the interpolated output value, the difference of the two values would have variance $v + v'$. We need to express v', the variance of an interpolated value at a random point inside a simplex whose corners have values with variance v.

That's the idea, but we must elaborate on it. First: how do we interpolate the samples in the training set? As the saying goes: "If you have a hammer, everything looks like a nail." We have an ALN learning system, and a proof that an ALN can represent any continuous function to any degree of accuracy on a compact domain. So if we approximate the training set outputs on the whole domain by overfitting an ALN as much as possible, then we will get something like interpolation of the output values of the points in the training set. Better would possibly be triangulating all the samples in the training set using a Delaunay triangulation, but we have found that overfitting has worked satisfactorily.

We start off by assuming that the overfitting has produced triangles in a two dimensional domain, or simplexes in higher dimensions. If n is the dimension of our problem, i.e. the dimension of the domain plus one, then the simplexes whose corners are inputs in the training set, have n corners. The outputs y_k of the samples (X_k, y_k) at the corners X_k have variance v, and an interior point $X = a_1X_1 + a_2X_2 + \cdots + a_nX_n$, where $a_k \geq 0$ and $\sum_{k=1}^{n} a_k = 1$, has an interpolated value $a_1y_1 + a_2y_2 + \ldots + a_ny_n$ with variance $v\sum_{k=1}^{n} a_k^2$.

If $a_k = 1/n, k = 1 \ldots n$ then the variance of the central point is v/n. For a one-input, $n = 2$-dimensional problem, we can compute the average variance v_2 in an interval (a simplex with $n = 2$ corners) as $v_2 = 2/3v$. We skip the details, but simple calculus shows that for an n-simplex, the average variance of a random point inside will be $v_n = 2v/(n + 1)$. If we then compute the variance v_{diff} using differences of output value for all the points in the validation set and interpolated outputs at the same point for the training set, we get $v_{diff} = v + 2v/(n + 1) = (n + 3)v/(n + 1)$. Now we conclude that the variance v of noise in the data is given by $v = (n + 1)v_{diff}/(n + 3)$ where v_{diff} is the mean squared error of the overtrained ALN on the validation set. What we refer to as the tolerance is the standard deviation of noise in the data. The tolerance has value $\sqrt{(n + 1)/(n + 3)}$ times the RMS error of the overtrained ALN on the validation set. We use the tolerance to stop pieces of the ALN splitting due to noise, i.e. if the RMS error on the training set points that belong to is is less than the tolerance. This is a major part of the strategy to avoid overtraining ALNs.

The above assumes that the level of noise is constant throughout the domain. Clearly if the noise is proportional to the value of the function we are trying to approximate, then we would take the logarithms of the output values to have a problem with a constant level of additive noise.

5. ALN inverses, global properties

The main reason for treating the output variable like the inputs in our representation is that an ALN which has ith weight components w_{ji} of all linear pieces j non-zero and of the same sign is strongly monotonic in x_i. An ALN for the inverse function w.r.t. that input is created simply re-normalizing the weight vectors of all linear pieces to be -1 on the new output x_i, and, if the relationship is increasing, interchanging maximum and minimum labels in the ALN. This simple inversion technique is discussed in Appendix C and has been applied in control of a mechanical system. For example, an ALN trained to give displacement as a function of force applied could be inverted to produce an ALN giving the force required to effect a specified displacement.

Properties of ALNs are derivable from bounds on the weights. For example if all pieces j of A have weight $w_{ji} \leq B$ then for two input vectors $\mathbf{s}$ and $\mathbf{t}$ which have all input components equal except the i-th and $t_i = s_i + c$ with $c \geq 0$, then $A(\mathbf{t}) \leq A(\mathbf{s}) + Bc$. Properties of ALNs like this enable one to impose $g$lobal conditions on an approximant A *a priori* by simply bounding the weights of linear pieces during training. Monotonicity, for example, can be forced upon an ALN approximant A in the case of a system whose ideal function f is known to obey certain physical or economic monotonicity laws.

6. Smoothing

To be able to analyze a smoothed ALN, its maximal deviation from an easily-analyzed PWLC function must be bounded. We introduce a smoothing epsilon $\epsilon_s > 0$. For a node $A = \vee_{\epsilon_s}(A_1, A_2)$ whose children, given input $\mathbf{x}$, compute values (including smoothing amounts) $f_1(\mathbf{x})$, $f_2(\mathbf{x})$ we let A compute the smoothed value:

$$f(\mathbf{x}) = \begin{cases} \vee(f_1(\mathbf{x}), f_2(\mathbf{x})) & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \geqq 4\epsilon_s \\ (f_1(\mathbf{x}) + f_2(\mathbf{x}))/2 + \epsilon_s + (1/16\epsilon_s)(f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \leq 4\epsilon_s \end{cases}$$

In the case of equal linear functions the smoothed value at a maximum node is greater than that value by ϵ_s. At places where $\mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid = 4\epsilon_s$, the quadratic fillet's values and first derivatives (w.r.t $f_1(\mathbf{x}) - f_2(\mathbf{x})$) match those of $\vee(f_1, f_2)$.

For a minimum node $A = \wedge_{\epsilon_s}(A_1, A_2)$

$$f(\mathbf{x}) = \begin{cases} \wedge(f_1(\mathbf{x}), f_2(\mathbf{x})) & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \geqq 4\epsilon_s \\ (f_1(\mathbf{x}) + f_2(\mathbf{x}))/2 - \epsilon_s - (1/16\epsilon_s)(f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \leqq 4\epsilon_s \end{cases}$$

.

If it is desired to have the second derivatives be continuous, in the case of $A = \vee(A_1, A_2)$ we can use a quartic fillet

$$f(\mathbf{x}) = \begin{cases} \vee(f_1(\mathbf{x}), f_2(\mathbf{x})) & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \geqq (16/3)\epsilon_s \\ \\ (f_1(\mathbf{x}) + f_2(\mathbf{x}))/2 + \epsilon_s + \\ (9/(128\epsilon_s))(f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 & \text{for} \quad \mid f_1(\mathbf{x}) - f_2(\mathbf{x}) \mid \leqq (16/3)\epsilon_s \\ -(27/(65536\epsilon_s^3))(f_1(\mathbf{x}) - f_2(\mathbf{x}))^4 & \end{cases}$$

.

One can show that f deviates from the maximum by at most ϵ_s everywhere. The fillet for a minimum node smoothed with a quartic is derived by changing the sign of what is added to $(f_1(\mathbf{x}) + f_2(\mathbf{x}))/2$. The quartic fillet for the $\vee$ is never less than the quadratic, but its width, in terms of $f_1(\mathbf{x}) - f_2(\mathbf{x})$, is $(\frac{32}{3}\epsilon_s)$ instead of $8\epsilon_s$. In terms of $\mathbf{x}$, the width of the fillet depends on the difference of slopes of f and g.

We emphasize that the ALN must use smoothing at every level because a node's intersecting child-ALN function graphs will have intersecting linear pieces. Futhermore, this requires that ϵ_s should be equal at all non-leaf nodes so fillets joining two linear pieces have similar properties everywhere. We have parts of the ALN approximant that are of high degree where overlapping fillets occur. We have expressed fillets as functions of the two inputs to a node, not in terms of the input space, because that is the minimal information that must passed up an ALN tree. If $f = g$ there is a special case where the fillet for a $\vee$ is $f + \epsilon_s$. This is used when leaf nodes split to give two new leaves identical to the original. Use of a standard least-squares algorithm such as SVD to adapt linear pieces would fail because that approach could never separate equal linear pieces from each other. For that to happen, some random choices are required as in our iterative approach.

We have not found a way to smooth nodes with more than two inputs rather than using several levels of two-input nodes. Also, more could be done to create smooth surfaces by carrying partial derivative information from the leaves up and down the ALN tree.

7. Properties of fillets

The fillets we have chosen have several properties useful in relating a smoothed ALN A_{ϵ_s} to the original piecewise linear ALN A. When overlapping fillets occur at different levels of the ALN at some point of input space, we can multiply ϵ_s by the number of non-leaf nodes on the path from that point to the root of the tree in order to get an upper bound on the maximum absolute deviation of a smoothed ALN A_{ϵ_s} from its PLCF A. This bound can be made more precise as follows.

If there are M active fillets on maximum nodes on the path from a leaf to the root of A_{ϵ_s} at $\mathbf{x}$, then $A_{\epsilon_s}(\mathbf{x}) \leq A(\mathbf{x}) + M\epsilon_s$, and if there are at most m minimum nodes on the path from $\mathbf{x}$ to the root, then $A_{\epsilon_s}(\mathbf{x}) \geq A(\mathbf{x}) - m\epsilon_s$. We could track which nodes have fillets that are active (encroached upon) for a given $\mathbf{x}$ and get an even better estimate. We can compute some values M and m for an ALN that work for all $\mathbf{x}$, starting with $M = m = 0$ at the leaf nodes and ascending recursively.

Another property is that the first partial derivatives of the filleted functions always stay between the corresponding slopes of the functions they are smoothing. Hence bounds imposed on the partial derivatives of the linear pieces of an ALN during training, are also valid for any smoothed ALN created from that one.

To see that, suppose B is an upper (or resp. b a lower) bound on all coefficients of x_i in all the linear pieces of smoothed ALNs A_1, A_2, and $A_{\epsilon_s} = A_1 \vee_{\epsilon_s} A_2$ or $A_{\epsilon_s} = A_1 \wedge_{\epsilon_s} A_2$. Then we can show recursively that $\partial A_{\epsilon_s}(\mathbf{x})/\partial x_i \leq B$ holds for all i and everywhere. The same argument shows that if b is a lower bound on the slopes of linear pieces, then $\partial A_{\epsilon_s}(\mathbf{x})/\partial x_i \geq b$ holds for all i and everywhere.

For the second partial derivatives, we have to assume bounds $c < C$ on them at a certain level and recurse to the next level up, starting of course with $c = C = 0$ for the linear pieces. Suppose we have bounds $b < B$ as above on the first partials of the linear pieces; then we get $\partial^2 A_{\epsilon_s}(\mathbf{x})/\partial x_i^2 \leq \frac{M}{8\epsilon_s}B$ for all i and everywhere, where M is the number of (active fillets on) maximum nodes on the path from the leaves to the root of A_{ϵ_s} at $\mathbf{x}$. $\partial^2 A_{\epsilon_s}(\mathbf{x})/\partial x_i^2 \geq \frac{m}{8\epsilon_s}b$ for all i and everywhere, where m is the number of (active fillets on) minimum nodes on the path from the leaves to the root of A at $\mathbf{x}$.

For quartic fillets the results are a bit more complicated, and are deferred to Appendix B.

8. Inversion of ALN functions strongly monotonic in an input

If a function y $= f(x_1, ..., x_{n-1})$ is strongly monotonic increasing in x_i, then there is a unique $g(x_1, ..., x_{i-1}, y, x_{i+1}, ..., x_{n-1})$ which computes the x_i. More precisely

$$g(x_1, ..., x_{i-1}, f(x_1, ..., x_{n-1}), x_{i+1}, ..., x_{n-1}) = x_i \tag{8.1}$$

and

$$f(x_1, ...x_{i-1}, g(x_1, ..., x_i, ..., x_{n-1}), x_{i+1}, ..., x_{n-1}) = x_i \tag{8.2}$$

for all vectors $\mathbf{x}$ in some subsets of $\mathbf{R}^{n-1}$ depending on the domains and codomains of f and g.

ALN are defined by expressions that have no limitations on the domains of inputs, which allows us to invert ALNs very easily. In this section we restrict consideration to non-smoothed ALNs. Some similar results can be obtained for smoothed ALNs, but only with a small error because fillets are evaluated always with respect to a certain output axis, and they don't survive a change of axis. Suppose ALN A has some positive number which is a lower bound on the coefficients w_{ji} of all linear functions j at the leaves, then the function computed by A is strongly monotonic increasing in x_i. Then the inverse ALN $A^{-1,i}$ with respect to x_i is computed by an ALN created from A by two simple changes.

1. All weights of all linear pieces are renormalized, i.e. each w_{jk} is multiplied by the reciprocal of the negative of the weight w_{ji} of the particular piece j, so that afterward, the weight w_{ji} becomes -1.

2. For the monotonic increasing case, all minimum nodes are changed to maximum nodes and all maximum nodes are changed to minimum nodes.

If the weights w_i of are all upper-bounded by some $B < 0$, then $A^{-1,i}$ can be created just using step 1. above. The node labels are not changed. Our usual representation is to put the output variable at the right of all the others, but this is just a convenience and destroys the nice symmetry in the case of inversions.

9. Exponential Smoothing

Exponential smoothing estimates the current mean of a sequence of real values. Specifically, if x_i is a sequence of reals, and F_i is the estimate computed after the i-th member of the sequence, then

$$F_i = (1 - \lambda)F_{i-1} + \lambda x_i$$

or equivalently

$$F_i = F_{i-1} + \lambda(x_i - F_{i-1}).$$

Here $\lambda \in [0, 1]$. The sequence x_i does not have to be stationary, and because of this, the choice of the value of λ is critical for tracking the mean in an accurate, timely fashion. If λ is close to 0, then the estimate will require many steps to obtain a good estimate of the mean. The advantage of a small λ is that it makes the variance of the smoothed estimate of the mean smaller than the variance of the source of the x_i. If λ is close to 1, then obtaining an estimate will be fast, but the estimate will have variance close to that of the x_i. The details will be given in Appendix D.

Our adaptive algorithm is based on using exponential smoothing to estimate the centroids, and several auxiliary values of linear pieces to which $\mathbf{x}$ belongs, based on the errors $A(\mathbf{x}) - y$, where y is the desired value of the output of a sample of data with $\mathbf{x}$ as the input. We also use exponential smoothing to decide whether the unknown function over a linear piece specified by some samples is generally convex-up or convex-down . Then if the piece splits into two, the operator which connects two new pieces will be a minimum or maximum, respectively. Samples $\mathbf{x}$ which are on fillets, have their effect on the exponentially smoothed values of the piece modified by the value of the membership function. The membership function is not zero or one; when fillets are present, it can be a high-degree polynomial in the $\mathbf{x}$ values belonging at least partially to the piece, so the change can be gradual and non-linear.

When a linear piece changes its centroid or weights, it is possible that the samples close to an edge change membership. The change in membership changes the values which influence the exponentially smoothed values associated with the piece, and the corresponding changes in membership of samples $\mathbf{x}$ in neighboring pieces change the values of the neighbors.

Thus learning involves a large number of coupled, non-linear, non-stationary stochastic processes.

10. Training of smoothed ALNs

Fitting linear pieces to data has been a solved problem for a long time, both using batch algorithms and iterative ones. Our problem is different. We must use an iterative approach because the set of data points belonging to a linear piece may change after any training step. More generally, a point may only partially "belong to" a linear piece if we use fillets. What counts in a training step is changing the coefficients of linear pieces to which a data point $\mathbf{x}$ during training at least partly belongs, in order to reduce the weighted mean of squares of differences of desired output values y from the function surface $A(\mathbf{x})$ computed by the current multi-level smoothed ALN A. We present proofs related to adaptation in appendix E.

All vectors in this section and its appendix will be n-dimensional column vectors unless otherwise stated. The reader will have to decide from the context how to treat the output variable of an ALN, as sometimes it will be there simply as an input to the ALN that goes unused in computing the ALN's output. Consider a matrix X of m rows of input vectors of dimension $n-1$. The i-th row will be denoted by $\mathbf{x_i}^T$. To each input vector, the training set assigns a desired real output y_i, and the m-vector of these will be denoted by $\mathbf{y}$.

Let $\mathbf{w} = (w_1, ..., w_n)^T$ be a normalized vector of weights, and $\mathbf{c}^T = (c_1, ..., c_n)$ be a vector that is intended to approximate the centroid of the data points belonging to a linear piece in all n axes. In particular, the value c_n, with training, should approximate the mean of the desired output values. These together generate an $n-1$-dimensional hyperplane (a "linear piece") which satisfies the equation $(\mathbf{x}^T - \mathbf{c}^T)\mathbf{w} = 0$. Our adaptation problem for the linear piece is to find $\mathbf{c}$,and $\mathbf{w}$ such that the linear piece, augmented by any fillets, has the least weighted RMS error from the given training points.

To take account of a possible fillet at input $\mathbf{x}_i^T$, instead of using y_i as the output (n-th) component of $\mathbf{x}_i$, which we would do for a linear fit, we use $r_i = y_i - (A(\mathbf{x}_i) - \mathbf{x}_i^T\mathbf{w})$. In this way, if a fillet goes above the linear piece, $A(\mathbf{x}_i) - \mathbf{x}_i^T\mathbf{w} > 0$, then the mean of the r_i will be less than the mean of the y_i by a corresponding amount and the centroid will be lower as a result, and will better reflect the centroid of the required linear piece. In a graph, the ALN *surface* with fillets sits at the centre of the data.

Adaptation involves the presentation of an input vector in a random row vector $\mathbf{x}_i^T$ of X and its corresponding y_i. The centroid $\mathbf{c}$ is adjusted iteratively, and in one step it becomes $\mathbf{c}'$ using an exponential smoothing technique to average the result over many training vectors. $\mathbf{c}' = (1-\lambda)\mathbf{c} + \lambda\mathbf{x}_i$, where $\lambda \in (0,1)$ is called the *learning rate*. As mentioned above, the output component of the centroid is computed differently $c_n' = (1-\lambda)c_n + \lambda r_i$. After a number of random training points have been presented, the exponentially smoothed values can be expected to stabilize. One advantage of using a centroid for each linear piece is that this can avoid some numerical problems of adding or subtracting large magnitude floating point numbers resulting in numbers of small magnitude.

Now we look at the more interesting problem of rotating a linear piece to better fit the data. To simplify notation, we assume below that the centroid of the data points belonging to the linear piece is 0, that is, the sum of the entries in the column vectors of y and X is 0. We developed an algorithm that statistically guarantees a reduction of the error by about a fraction λ for each pass through the data points. This will become more precise in the later proofs.

We exponentially smooth each component j of the input vector by letting $d_j' = (1-\lambda)d_j + \lambda x_{ij}^2$ for each j = 1,...,n-1. d_j tends towards the mean value of the squares of the x_{ij}. Then for each training point x_i we let the components of w change by adding a fraction of x_i as follows:

$$w_j' = w_j + \lambda x_{ij}(y_i - A(x_i))/d_j. \tag{10.1}$$

Assuming that the ALN A is just a linear piece, $A(\mathbf{x}_i) = \mathbf{x}_i^T\mathbf{w}$, after this step $y_i - \mathbf{x}_i^T\mathbf{w}' = (y_i - \mathbf{x}_i^T\mathbf{w})(1-\lambda)\sum_{j=1}^{n-1}(x_{ij}^2/d_j)$. If x_{ij}^2 is close to the mean value d_j over i, then $y_i - \mathbf{x}_i^T\mathbf{w}' \approx (1-\lambda)(y_i - \mathbf{x}_i^T\mathbf{w})$. Thus the magnitude of the error at the training point i can be expected to decrease by a fraction near λ. Simple examples show that the new linear piece may have increased error at other points. In Appendix E we give a plausibility consideration that the root mean square error $\|(y - X\mathbf{w})\|$ decreases by a fraction near λ when all training input vectors have been presented.

11. Adaptation by splitting

After enough steps of adaptation of linear function coefficients at the leaves of an ALN have occurred, the rate of change slows. This is close to some local minimum of RMSE for the current ALN tree. To reduce error further, the tree grows by splitting some leaf nodes into two which have RMS errors on the training data which exceed the target global RMS error. To this end, the tree is scanned and statistics are accumulated for each leaf regarding how often it is involved in adaptation steps during a single pass through the training data and to what total extent it is responsible. A node may be only partially resposible for the output value for a given input to the network if there is a node above it which shares the responsibility between its children on account of smoothing.

Necessary for splitting a leaf with error larger than prescribed are several additional criteria which must be met during one pass through the training data:

- The number training sample points whose value is determined by the leaf must be large enough to make it unlikely that either new leaf will become under-determined.
- The total responsibility of the leaf must exceed a multiple, e.g. four times, the dimension n of the problem (which also implies the same lower bound on the number of training sample points).
- A curvature measure must be sufficiently different from noise so it is clear whether the data belonging to the leaf is more convex-up or convex-down, hence requiring that the leaf be replaced by a minimum or resp. maximum node with two child leaves, each initially equal to the one being split.
- Failing clarity under the previous item, a non-root node splits into a maximum or minimum the same as the parent of the leaf being split, and the root splits in a random way.

The first two conditions assure that there is a good chance for each of the two child leaves after a split to have enough points (n) and responsibility associated with each to be determined or over-determined. We now define a curvature condition. Generally the idea is that the distance of data samples belonging to a piece are analyzed to find out whether they are greater or less than noise near the centroid of the piece or near the periphery. A problem arises in higher dimensional spaces that the volume of spherical shells grows outward from the centroid as the $n-2$ power of the distance. This means that randomly generated input points will tend to be located far from the centroid. Our answer to this is to note that all we need is curvature in a *single* direction, the direction along which the linear piece can split with a good chance of decreasing the error. Hence our measure of curvature is an average of measures along the $n-1$ axes of the input space. Suppose f represents the sample output values belonging to an affine piece A. Our measure of convexity of data points on the piece along axis j is C_j, which is obtained by exponentially smoothing $f(x_1,...,x_{n-1}) - A(x_1,...,x_{n-1})$ over all training points of the piece which are further away from the centroid than the standard deviation of all points along axis j. The units of the C_j's are the units of the value of f. The algorithm averages the values of C_j for all axes and chooses to split as a maximum node the if the average is positive and larger than the prescribed noise level. This means the function f tends to have larger values than the piece A near the outer rim (the graph of $f-A$ is shaped like a bowl). The goal is not to determine convexity per se, but rather to determine how best to continue reducing the error of the fit by splitting a piece into two.

As an added refinement, we can modify this criterion in order to keep the function surface simple when C_j is within the level of noise. Then we look at the maximum or minimum of the parent node, if it exists in the ALN tree, and choose the same for the node being split. There is room for further refinement of this technique, but at this moment, the implementation actually produces an overwhelming preponderance of maximum nodes for convex-down paraboloids. Further perfecting the analysis of curvature in the presence of noise could lead to being able to better analyze qualitatively a function surface, even in high dimensions, by looking at the synthesized ALN tree.

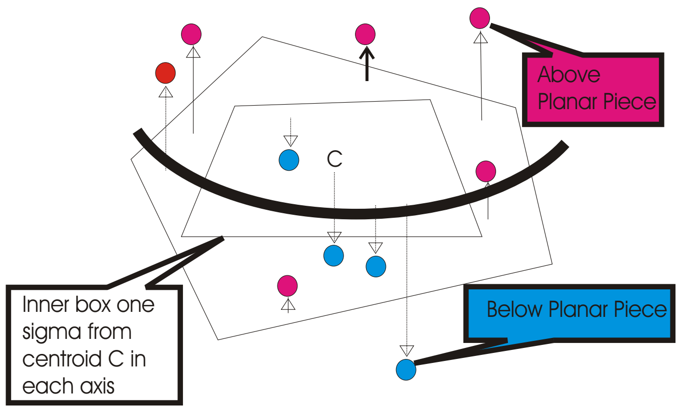


Figure 3. The preponderance of data points above the ALN piece outside the rectangle suggests a maximum node split (convex down heavy curve).

12. Taking the work out of evaluation

A complex surface may require thousands or even millions of linear pieces to fit it to the desired accuracy. We have examples where using fillets can achieve the same accuracy with a small fraction of that number of pieces, but still there is a lot of work to do to evaluate the ALN at a point. The amount of computation can be greatly reduced by partitioning the input space (which we here assume is a rectangular $n-1$-dimensional block into a number of such smaller blocks. On each of the subblocks, only a subset of the linear pieces are ever needed to compute the output of the ALN even with fillets. To eliminate some subtrees of the ALN tree from those required on the subblock, a careful analysis of the maximum and minimum values of those subtrees on the subblock is required. If two subtrees which meet at a minimum (or resp. maximum) node have ranges of values which don't intersect, then only the subtree with lesser (or resp. greater) values is useful on the subblock. Generally it is possible to make subblocks so small that only n linear pieces are active on the subblock. The exceptional case is one where more than n linear pieces meet at a single point (e.g. think of a pyramid with four sides for $n = 3$). In that case we could still achieve the result of having only n linear pieces per subblock if we "file off" the point on the pyramid and put a flat piece on the top. The error introduced by the change would have to be less than the allowable error in fitting the data.

13. Statistical theory and applications of ALNs

ALNs and similar techniques are important in statistics, including such applications as regression, time series analysis, classification, and function approximation.

Statistical theory also exists for estimation techniques very closely related to ALNs, indeed identical for the case where smoothing fillets are negligible. In regression, models are estimated from existing data in order to predict future values of the dependent variable, y, given values for the independent variables x where in general x is multivariate; in this context, the importance of variables in x is also of interest. In time series analysis, useful models may involve regressing current on lagged values of y. Models are sometimes used to classify observed data into one of several population types, such as product purchasers, solvent companies, fraudulent behavior, speech recognition, military targets, terrorist recognition, medical diagnoses, and states of complex system control. In this context, false positive and negative model predictions are important due to safety and cost concerns. In function approximation, the estimated model response y as a function of x is used to better understand the properties of the true underlying function and perhaps the process generating the data; in this case, interpretability of the estimated model is important.

In this section, we emphasize ALNs for their interpretability, the ease with which variable importance may be measured, and their openness and lack of hidden surprises. We also mention related techniques which are piecewise linear without smoothing fillets to better understand the wide applicability of ALNs in statistics and the existence of statistical theory.

ALNs may be used in nonlinear statistical estimation when the exact nonlinear model is unknown, the usual situation. With enough pieces, any nonlinear function may be approximated quite well. Appendix A shows that ALNs are plausibly universal approximators. We say, plausibly, because of the difficulties handling smoothed fillets. However, there is a large body of statistical theory, Stochastic Approximation (SA), which deals with iterative statistical estimation techniques, of which ALNs are one example. The origins of SA are due to Robbins and Monro (1951) and Kiefer and Wolfowitz (1952). SA may be viewed as a stochastic counterpart of iterative numerical analysis, and its iterations are equivalent to those of ALNs without smoothing fillets. Under fairly general conditions, estimators arising from SA are consistent and even asymptotically normally distributed. Again, plausibly, the minor influence of smoothing fillets in ALNs should not alter these properties. We should also mention that there is statistical theory for piecewise linear regression models. Basically, asymptotic theory is available which may be relied upon for large samples, resulting in standard normal tests for coefficients based on the use of indicator variables for hinges. Distribution theory for the hinges themselves, which are ratios of coefficients, is available but more complicated. Feder (1975a, 1975b), Hinkley (1969, 1971), Quandt (1960), and Velilla (1998) are useful references to this theory. This theory also suggests that if the fillets in ALNs are negligible, ALN estimators will be consistent and asymptotically normal.

There are a couple of techniques that are similar, but deficient to ALNs. Friedman (1991) proposed multivariate adaptive regression splines (MARS) which is a sum of Min and Max operators where each linear term is univariate. MARS is therefore similar to ALNs in its use of Min and Max operators, but deficient since the terms are univariate rather than multivariate as is the case with ALNs. It is not a universal approximator, but is widely used in data mining where large data bases and many variables are frequently found. Obviously, any application for MARS could benefit from the use of ALNs on the same data. Breiman (1993) proposed an iterative algorithm for estimating any number of multivariate linear pieces joined together as a sum of Min and Max operators which he termed hinging hyperplanes (HHP). In theory, HHPs are universal approximators for any multivariate function. His algorithm is guaranteed to produce a local, not global, least squares solution. Given any fixed HHP model, an equivalent ALN model may be derived, and vice versa. However, the structure of an ALN permits far faster computation. Also, the presence of fillets in ALNs prevents estimation of local rather than global solutions. Again, any application of HHP could benefit from the use of ALNs.

We should mention an important subfield of statistics, time series analysis, which can also use ALNs. One of the standard approaches to time series analysis and forecasting is the use of Autoregressive Integrated Moving Average (ARIMA) models developed most fully by Box and Jenkins (1976). A simple autoregression is a multiple regression on lagged values of the time series and may be inverted into an equivalent moving average. All of our discussion therefore applies to time series analysis. In fact, various authors have explored special cases of piecewise linear models in time series analysis, such as Hansen (1997), Medeiros et al. (2002), Proietti (1998), Tong (1983, 1990), Tong and Wu (1982), Hamilton (1989), and Tsay (1998). Applied to time series, ALNs are much more general.

Since ALNs are an extension of simple piecewise linear models, potential applications in statistics are limitless. Any application of piecewise linear estimation can be potentially improved by the use of ALNs. A Google search using key words like piecewise linear regression, hinge regression, two phase regression, MARS, Hinged Hyperplanes etc. will turn up thousands of applications. We mention a few specific applications to conclude this section. Shaban (1980) gives an annotated bibliography for the field of hinges and two-phase regression, citing theoretical and applied work. Accounting and Auditing Fanning and Cogger(1995) compared ALNs with standard discriminant analysis techniques for the detection of management fraud using information that would be available to auditors (but not generally available to the public). They found both techniques useful, with ALNs superior in predictive ability. The natural extension of this study is found in Fanning and Cogger(1998) where only publicly available information was used. Again, they found ALNs superior in detecting management fraud. Naturally, a next step would be to combine both publicly available and privately available audit information for fraud detection.

2.2 Marketing

Allenby (2004) develops a choice model for packaged goods. In the presence of discrete purchase quantities and quantity price discounts, piecewise linear models are found to be optimum. Lee and Brown (1985) examine the use of discount coupons for the purchase of frozen orange juice. They find the best model for estimating demand is piecewise linear. Suits (1955) also studies a fruit market, that of water melons, and finds an econometric model incorporating piecewise linear components. Meullenet and Xiong (2004) find a piecewise linear model best describes marketing data useful in developing the preference mapping of cheese sticks. All of these studies could be improved upon by using more general ALNs.

Luce and Tukey (1964) developed Conjoint Analysis. This statistical procedure has been widely used in Marketing, with many thousands of studies conducted. The basic idea is monotonic analysis of variance, where individuals express their preferences for various product profile descriptions and rank them. Then, part-worth functions are estimated to be as consistent as possible with overall rankings.

Conjoint Analysis can be thought of as an analysis of variance problem in which the attribute levels are coded by indicator variables and the goal is to find a monotonic function of the input data that minimizes error. Frequently, the monotonic function is approximated by a piecewise linear function. ALNs could greatly improve Conjoint Analysis.

Data mining is a relatively new field, but has attracted much attention in Marketing especially where scanned consumer characteristics are involved. In recent years, the availability of software for handling such data sets with minimal functional assumptions has led to an explosion of applications. Of note here would be the ease with which ALN technology can handle large data sets.

Organizational Behavior

Charnes, Cooper, and Rhodes (1978) developed Data Envelope Analysis (DEA). It is the technique of choice for studying productivity and efficiency of organizational units such as corporations, corporate divisions, small work units, and even economies as a whole. So-called efficient frontiers in DEA are inherently piecewise linear, and their statistical analysis of necessity uses such functions. See Hirschberg and Lye (2001). For a complete annotated bibliography of DEA, see Seiford (1995).

Finance

Fanning and Cogger (1994) used ALNs for the prediction of corporate bankruptcy. As in the previously cited Accounting applications, ALNs were superior to standard approaches such as linear and quadratic discriminant analyses. Cogger, Koch, and Lander (1997) studied the behavior of international equity markets during the months surrounding the volatile 1987 fluctuations. ALNs were uniformly superior to standard regression models across countries and across time. Bracker, Cogger, and Fanning (2001) developed successful futures options trading strategies based on piecewise linear models and found statistically significant risk adjusted rates of return for these strategies involving currency futures options for a variety of countries.

Brach and Wang (1995) found piecewise linear models best for describing risk arbitrage performance when stock swap offers were constrained by collars. Dantzig and Infanger (1993) examined portfolio optimization using multi-stage stochastic linear programming. Their models incorporated several piecewise linear approximations. Konno and Yamazazaki (1991) also studied portfolio optimization within the context of the Tokyo stock market. Employing a mean-absolute deviation criterion, instead of the usual mean-variance criterion, they found piecewise linear functions essential to their approach. Mathiesen (2006) has published an on-line survey of empirical studies in Finance of ownership structure and stock performance. A number of these studies, not specifically referenced in the present paper, can be found at

`www.encycogov.com/A5OwnershipStructures/OwPerfStudies/Table_Ow_HIJ.asp`

.

An example of such a study is Morck, Shleifer, and Vishny (1988). Ritchken (1985) studies the pricing of stock options and finds piecewise linear models useful in the analysis of pricing bounds. All of these applications could benefit from the use of ALNs as a more general approach.

Robert F. Engle was awarded the 2003 Nobel Prize in Economics for methods of analyzing economic time series with time-varying volatility (ARCH). In his original paper, Engle (1982) states, it is likely that other formulations of the variance model may be more appropriate for particular applications.simple alternatives areabsolute value forms. The absolute value function, of course, is one of the simplest piecewise linear functions. Later, Bollerslev (1986) extended this model to GARCH which included a quadratic term. Recently, Engle (2002) suggested that the GARCH model might be improved upon in some situations with the inclusion of a piecewise linear term. So, 20 years after his seminal and Nobel Prize winning work, Engle is still suggesting piecewise linear functions for modeling time-varying volatility. We might here suggest combining Engles recent suggestion with his suggestion in his 1982 paper: include not only the indicator function on h, but replace the quadratic y with its absolute value or a more general piecewise linear function. Economics There are a number of publications in economics reflecting underlying piecewise linear budget constraints. These include Zhan (1999), Moffitt (1986), Friedberg (2000), Harvey (2001), and Heim and Meyer (2003). Reiss and White (2005) use piecewise linear models for the estimation of demand elasticity when pricing is nonlinear. Again, ALNs would be useful to consider. Clive Granger, the co-Nobel prize winner in Economics for 2003, has also written extensively on the use of nonlinear models in Economics. See Granger and Terasvirta (1993). While discussion of piecewise linear models is largely absent from Grangers work, it is not difficult to extend many of his models to that framework and, of course, to ALNs.

Cooper (1998) studies the possibility of multiple regimes in U.S. output fluctuations. Such regime changes may be due to oil price shocks, the outbreak of war or other conflicts, acts of terrorism, etc., which make time series susceptible to structural change, making the data in different time periods incompatible. For example, the post-1947 period is often split from the pre-war period in studies of output for the U.S. economy. Other economic variables have also produced evidence of structural change. Merz (1999) has studied unemployment flows and indicates that the sampling period does matter. Cooper used Classification and Regression Trees, CART, due to Breiman et al. (1984) which are inherently piecewise linear, and Merz used a Markov Switching Model, for which ALNs can be substituted. Fair and Jaffee (1973) used piecewise linear models to study markets in disequilibrium. Not repeated as far as is known, this study or others like it could shed light on short term market behavior and ALNs would be a more general framework for such studies.

Engineering Attoh-Okine (2000) has utilized ALNs for modeling highway pavement roughness predictions. He has used both ALNs and MARS in his studies and found, not surprisingly, ALNs superior. Additional studies along these lines are in Attoh-Okine and Roddis (2004), and Attoh-Okine and Cogger (2009).

Medicine There are many examples of the use of piecewise linear models in medical practice. Physicians and some patients are familiar with them. Blood pressures above 140/90 (Systolic and Dyastolic) lead to treatment recommendations for hypertension. PSA (Prostate Specific Antigen) levels above 3.0 lead to further invasive tests (biopsies) for prostate cancer in men or breast cancer in women. Lymphocyte counts above a threshold or

14. Applications of ALNs to forecasting in the energy industry

ALNs have been used successfully in the electrical power industry both for short-term demand (load) forecasting and for forecasting the amount of wind power produced by a large number of wind turbines. ALNs were entered into a 2002 demand forecasting competition by David Esp of National Grid Transco PLC. The competition, which was open to anyone in the world, was sponsored by EUNITE (The European Network on Intelligent Technologies for Smart Adaptive Systems). An entry from Taiwan came in first using the Vector Support Machine method. Esp's ALN entry came in a close second with about 2Prediction 1 hour ahead: 0.80% Prediction 24 hours ahead: 1.17% Prediction 48 hours ahead: 1.40%

E.on, a large electrical power and natural gas company asked W. W. Armstrong to develop a wind power forecasting system for thousands of wind turbines feeding power into its system. Wind speed forecasts from the Deutsche Wetterdienst (German Weather Service)sent by email to the Power Plant Dispatching department at E.on, giving hourly forecasts up to several days in advance. The total production of power from wind was available at 15 minute intervals. Data was not available that could relate the wind speeds at particular weather stations to the production of power at individual turbines or wind farms, so the system was clearly suboptimal. Nevertheless, the results were good enough that they were used in commercial operations for several years.

15. Open-source software

The source software is at GitLab.com after being hosted on Source Forge and Gitorious until those sites passed the task on to a successor. It is available to developers in the form of a GIT repository. The software is written in C and C++ and runs with no changes on Microsoft Windows (R)computers using the Visual Studio C++ development environment. Conversion to other platforms should be straightforward using the source code. The source also includes the ALNfit Pro application which offers and easy way to test ALN learning by giving a data file with inputs and desired outputs for training (or inputs only, for testing). Unlike similar systems ALNfit Pro first determines the level of noise in the data, assuming that each data sample's output has identical and independent noise. It uses this level to stop enlarging the network by splitting leaf nodes. Again unlike comparable systems, ALNfit Pro allows one to impose a priori conditions on the trained result, such as monotonicity, or restriction of partial derivatives in a certain variable to a given range. Another feature deals with missing values in a datafile; specifically one can use ALN learning to estimate a missing value from data where it is present. One can iterate the process for missing values in several columns of a datafile.

Appendix A. Universal approximation, qualitative properties of fit

We now give a proof of this. By continuity of f, there is a lower bound m on the values of f on D. Let $\mathbf{x} \in D$. Again by continuity, there is an open neighborhood of $\mathbf{x}$ wherein the values of f deviate from $f(\mathbf{x})$ by less than ϵ. We can choose a rectangular box which is the product of intervals on the $n-1$ axes inside this neighborhood and symmetrically placed around $\mathbf{x}$. We take a copy B_1 of this box and place it at level m, and we take another copy B_2 shrunk to half size around $\mathbf{x}$ at level $f(\mathbf{x})$. Now we construct an ALN with pieces that form the top and sides of the truncated pyramid indicated by the boxes. All the pieces are connected by $\wedge$-nodes to form a convex-up bump. This ALN is everywhere less than $f+\epsilon$. In the interior of the box B_2, it is greater than $f-\epsilon$. Because it is compact, D can be covered by a finite number of these interiors. Then the ALN formed by combining the ALNs of the boxes in the cover using maximum nodes approximates f as required.

This existence proof constructs a very large ALN compared to what is necessary. Ideally, convex-up parts of the function surface of f would be fitted by convex-up ALNs formed by $\wedge$-nodes, and similarly for convex-down parts using $\vee$-nodes. Simple saddle-surface parts would be fitted with ALNs using both $\wedge$ and $\vee$ operators, one kind above the other in the tree. A good ALN approximation would thus reflect some of the qualitative properties of f. How to achieve such an ideal fit is an open question. It may help to think of convex-up (or resp. -down) ALNs as "hammers" with which one can augment an existing ALN by adding a maximum (or resp. minimum) node at the root. We can thus deform a hyperplane or other ALN surface, like we are doing metalwork on a car's exterior, to fit f better. It would be nice if some adaptation algorithm could found to produce an ideal qualitative description of the surface described by given data, but that is not the case with our algorithms. At least this metalwork analogy shows that a multilevel tree of maximum and minimum nodes is appropriate: imagine a flat sheet of metal and hit it from below with a large hammer with a gently rounded head. Then take a smaller hammer and put a downward dent in the bump so formed. Then take a still smaller hammer and put a smaller upward dent in the downward pointing one. This can be continued to any level.

Appendix B. Proofs about fillets

We consider quadratic fillets.

Suppose B is an upper bound on all coefficients on x_i in ALNs A_1, A_2, and suppose $f(\mathbf{x}) = A_1(\mathbf{x})$, $g(\mathbf{x}) = A_2(\mathbf{x})$. Suppose $\partial/\partial x_i f(\mathbf{x}) \leq B$, $\partial/\partial x_i g(\mathbf{x}) \leq B$. Then for the quadratic fillet on $f \vee g$ we have for the region where $\mid f - g \mid \leq 4\epsilon_s$, i.e. for the place where the fillet replaces the original functions:

$\partial/\partial x_i \left[(f+g)/2 + \epsilon_s + (1/16\epsilon_s)(f-g)^2\right] =$

$[1/2 + (2/16\epsilon_s)(f-g)]\, \partial f/\partial x_i +$

$[1/2 - (2/16\epsilon_s)(f-g)]\, \partial g/\partial x_i$

Because of the bounds on $f - g$, the expressions in square brackets are non-negative, so we get an upper bound by replacing the partial derivatives by B. Hence the fillet satisfies

$\partial/\partial x_i \left[(f+g)/2 + \epsilon_s + (1/16\epsilon_s)(f-g)^2\right] \leq B.$

Where the fillet has not replaced $f \vee g$, the bound still holds, so it holds everywhere. Hence by recursion starting from the leaves of A_s, $\partial A_s(\mathbf{x})/\partial x_i \leq B$ holds for all i and everywhere in the domain. Suppose f and g are two ALN produced functions entering a node with a quartic fillet, giving rise to a function h. It is sufficient to consider a maximum node, to consider the case of functions of one input, and to set $\epsilon_s = 1$. Let B be an upper bound on f' and g'. We have to show B is also a bound on h'.

We have $h' = [1/2 + (18/128)(f-g) - (108/65536)(f-g)^3]f' + [1/2 - (18/128)(f-g) + (108/65536)(f-g)^3]g'$. Taking the derivative, we find that the extrema of the coefficents of f', g' are occur where $(f-g) = 16/3$, hence $|(18/128)(f-g) - (108/65536)(f-g)^3| \leq 1/2$. So the coefficents are both non-negative and we can upper bound h' by substituting B in place of f' , g'. Thus $h' \leq B$. QED For a lower bound b on f', g' we get $h' \geq b$" Considering now the second derivative of h, we get $h" = [1/2 + (18/128)(f-g) - (108/65536)(f-g)^3]f" + [1/2 - (18/128)(f-g) + (108/65536)(f-g)^3]g" + [(18/128) - (324/65536)(f-g)^2](f'-g')^2$. For C an upper bound on f" and g", we get $h" \leq C + (9/8)B^2$, and for c a lower bound on f", g" we get $h" \geq c - (9/8)B^2$.

Appendix C. Inversion

Let's take the strongly monotonic increasing case first. If A is a leaf, then the result follows from step 1. Now suppose $A = A_1 \vee A_2$ or $A = A_1 \wedge A_2$. Let y be any real number. Suppose we have proved for all ALNs of lower maximal path length from leaves to root than A that the above inversion methods hold. Then define $x_1 = A_1^-1(y)$ and $x_2 = A_2^-1(y)$. Suppose $x_1 \leq x_2$, otherwise interchange A_1 and A_2 to make this hold.

We have $A_1(x_2) \geq A_1(x_1) = y = A_2(x_2) \geq A_2(x_1)$ so $y = A_1(x_2) \wedge A_2(x_2)$ where $x2 = A_1^-1(y) \vee A_2^-1(y)$. Moreover $y = A_1(x_1) \vee A_2(x_1)$ where $x1 = A_1^-1(y) \wedge A_2^-1(y)$. Since these expressions hold for all y, we get the result by recursion on maximal path length from leaves to root. The case for strongly monotonic decreasing is similar.

Appendix D. The variance of exponentially smoothed values

Let x_1,x_2,x_3,... be a sequence of i.i.d. random variates with mean μ and variance σ^2. Let $0 < \lambda \leq 1$, and let F_1,F_2,F_3,... be a sequence of smoothed values which are computed according to $F_i = (1-\lambda)F_{i-1} + \lambda x_i$. We set F_0 to an arbitrary value to start. Here we set it to zero. We now want to determine the variance of F_i in the limit as $i \to \infty$.

Now suppose that we have equilibrium such that F_i and F_{i-1} have the same distribution, then we can consider the past sequence of inputs to be infinite without changing that distribution and write

$$F = \lim_{j\to\infty} \lambda \sum_{i=1}^{j} (1-\lambda)^{j-i} x_i. \tag{D.1}$$

Then for the mean we have

$$mean(F) = \lim_{j\to\infty} \lambda \sum_{i=1}^{j} (1-\lambda)^{j-i} \mu = \mu \tag{D.2}$$

and for the variance we have

$$var(F) = \lim_{j\to\infty} \lambda^2 \sum_{i=1}^{j} (1-\lambda)^{2(j-i)} \sigma^2 = \frac{\lambda}{2-\lambda}\sigma^2. \tag{D.3}$$

In machine learning, we often look at the deviation from μ as a root mean square (RMS) error, which equals the standard deviation of the mean. Then, for example, if $\lambda = 0.05$, the exponentially smoothed value will have standard deviation about 16% of that of the sequence of x_i. A smaller λ averages the inputs in such a way as to further reduce the noise, at least in the limit of an infinite sequence.

Now we consider the consequences of using a truncated sequence. One can show that the truncated sequence which gives F_i is a biased estimator of the mean of the x_i. An unbiased one would be $F_i/(1-(1-\lambda)^j)$, whose variance is equal to the variance of the series with infinitely many terms, times $1 + (1-\lambda)^j$. If we take, for example, $\lambda = 0.05$ and j=10, the variance of the unbiased estimator is only 60% greater than the estimator with infinitely many terms in the series. With 20 terms, this drops to 35%. Since in our application the process which generates the x_i is non-stationary, there is no way to choose a value for j.

For a given j, reducing λ generally decreases the bias in F_i and causes the time for its variance to get below a certain threshold to increase.

Appendix E. Proofs about adaptation

We analyze the average effect of adapting to one of the training points in [Xy]. Passing through all training vectors is often referred to as one *epoch* of training. The w, c, and c_n values change after each training point, and even the set of points in [Xy] changes, which means we can't model a whole epoch this way. We don't have a proof that error decreases for the actual algorithm, but this proof suggests that we can expect the adaptation of linear pieces to converge to something with low root-mean-square-error (RMSE). This may happen even faster than predicted below, since the changes happen on a per-point basis.

Let S be an $n-1 \times n-1$ matrix with 0 for all off-diagonal entries and with (j, j) entry equal to $((1/m)\sum_{i=1}^{m} x_{ij}^2)^-1/2$, where x_{ij} is the j-th component of the i-th row vector in X. Let $V = XS$. In effect, V is a matrix of "normalized" input vectors where each component of all training inputs is scaled to have unit variance. Suppose f_i is the current smoothed ALN output for input x_i, then let $u_i = y_i - f_i$. In the case of a sole leaf ALN $u = (y - Xw)$. Now we compute the average weight change of w over all training points as follows: $w' = w + (1/m)\lambda SV^T u$.

Then at the beginning of the step, the sum of squared errors $E = u^T u = \|u\|^2$. After the step, the values of u change by the amount the linear piece has moved, which depends on the above change in the weight of the piece from w to w', applied to the vectors of X: $E' = \|u - (1/m)\lambda XSV^T u\|^2 = \|u - (1/m)\lambda VV^T u\|^2$. Expanding this, we get $E' = E - (2/m)\lambda u^T VV^T u + (1/m^2)\lambda^2 uVV^T VV^T u = E - (2/m)\lambda\|V^T u\|^2 + (1/m^2)\lambda^2\|VV^T u\|^2$. Since the second term is generally negative, the new error will be less as long as we can upper-bound $\|VV^T u\|$. $VV^T u$ is an m-vector formed by adding the columns of V after multiplying them by the respective components of the n-vector $V^T u$. The L2-norm of each of the columns of $VV^T u$ is thus the absolute value of the corresponding component of $V^T u$, and by the triangle inequality, the norm of the sum of these vectors $\|VV^T u\|$ is less than the L1-norm of $\|V^T u\|$. By a known relationship between the norms we get $\|VV^T u\| \leq \sqrt{n}-1\|V^T u\|$. Hence $E' \leq E - ((2/m)\lambda - (n-1)/m^2)\lambda^2)\|V^T u\|$. We see that $E' \leq E$ if $\lambda \leq 2/(n-1)$. The case where the error doesn't decrease is when $V^T u = 0$, but this is the criterion for the sum of squared errors E being minimal over all w. In that case, $w' = w$.

We can improve on this by noting that for many data sets the columns of V are almost orthogonal. This is because each component of a random x_i has mean 0 by assumption, the normalization in V makes the standard deviation unity, and an assumed lack of correlation between input columns will produce a regression coefficient close to 0, at least for large m. This won't be the case if one input component is close to being a multiple of another one. In cases like this, the input data points don't fill a volume in n-space but rather occupy a region close to a lower dimensional manifold embedded in (n-1)-space.

If we do assume the columns are orthogonal, then VV^T is an $(n-1) \times (n-1)$ orthogonal matrix, the triangle inequality is not needed, and we have $E' = (1 - (1/m)\lambda)^2 E$. T

If we now look at the situation we can expect after an epoch of training, and using the root mean square errors R, R', we get $R' = (1 - \lambda)R$. This justifies our "rule of thumb" that in general, an epoch of training reduces the RMSE of the fit roughly by about a fraction λ.

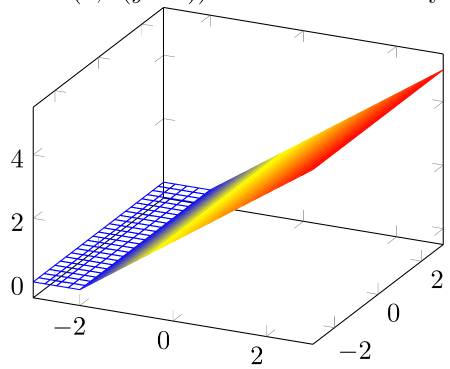
The above proof has many assumptions in it that don't hold in practice, so we can't claim we have a "convergence theorem". In actuality, we have not a rehearsal of separate steps, but an "all-singing, all-dancing" performance of training points, coefficients of linear pieces and fillets that needs further study. We believe we have given here a plausibility argument that the adaptation algorithm generally decreases the sum of squared errors over the whole surface as adaptation progresses with a learning rate that doesn't have to be infinitesimally small. In most cases, $\lambda = 0.2$ is satisfactory, but once splitting of linear pieces has stopped, an "ALN polishing" phase with $\lambda = 0.05$ may improve the accuracy of the fit.

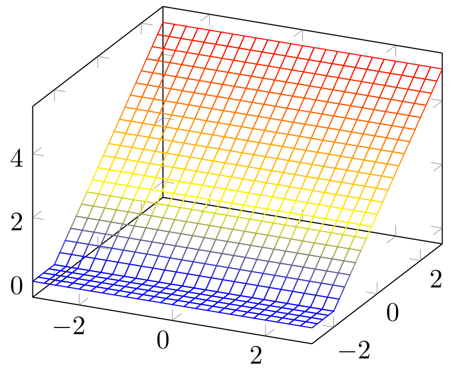
Splitting of linear pieces to grow the ALN doesn't increase the sum of squared errors, since the two pieces have the same coefficients as the original one, so a decrease of RMS error can be expected over the whole process of tree growth as well as adaptation. The decrease is not perfectly monotonic because of noise.

Appendix F. Other systems for fitting functions with affine pieces

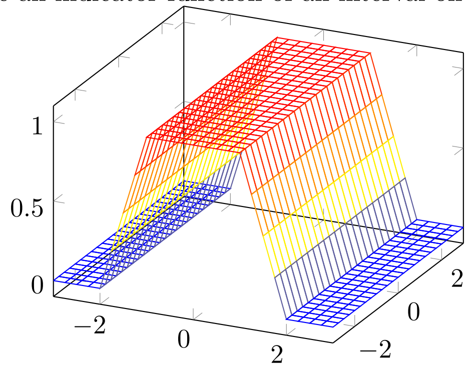
The most notable system based on fitting function graphs with hyperplanar pieces is MARS, originally described by Friedman and now embodied in several products, notably a proprietary version by Salford Systems Inc., and another system developed by Leo Breiman called hinged hyperplanes. We limit ourselves here to the discussion of functions of one or two inputs, since that will show some important properties of the methods.

MARS is based on recursively partitioning the domain of the function into various sized rectangles, except that a rectangle is not represented by an indicator function but rather by a function like it made up by multiplying by constants, adding, subtracting, or multiplying functions of the following types: $z = max(0, a(x - t))$, $z = max(0, a(y - t))$ where a can be any real constant.



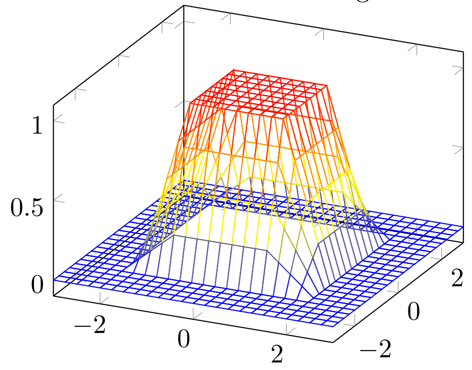


By adding or subtracting several functions like the first one we can get something like an indicator function of an interval on the $x - axis$:



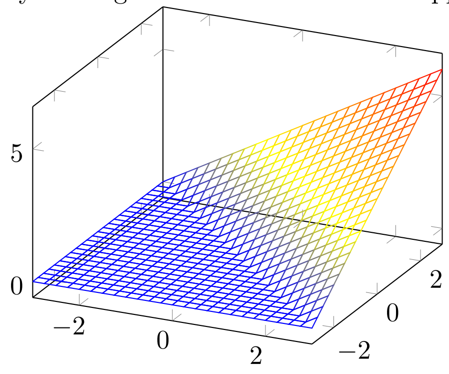
One can make a similar function, possibly with a slanted top, and if we use a large sum, we can approximate any continuous function on a closed, bounded interval uniformly to any accuracy.

Then we can do the same for the $y - axis$ and get close to an indicator function for an interval there. Finally, multiplying the two together gives us something like the indicator function of a rectangle.



By adding and subtracting real-valued multiples of smaller, translated bumps, it is rather evident that one can approximate any continuous function. The rigorous proof that the procedure can approximate any continuous function on a compact domain uniformly to any desired accuracy follows from the Stone-Weierstrass theorem. There are other refinements to the MARS process, such as smoothing the corners at hinges by cubic polynomials, that can greatly reduce the number of basis functions that have to be combined to fit a smooth function.

There is a problem with the MARS method though, and that is difficulty in fitting a function like $z = max(0, x + y)$, a hinge which is not parallel to either axis. It is possible, but the approximation may need a huge number of approximations in tiny rectanges that cross the hinge. Even on rectangles that don't cross the hinge, using a product of affine functions to try to approximate a function like x+y gives only an approximate result. For example $(1 + x)(1 + y) - 1 = x + y + xy$ differs by xy which is small in a small rectangle about (0,0) but still justifies saying that many rectangles will be needed for an approximation.



Here is a MARS result that used 23 basis functions. It is seen edge-on so as to make more visible the slight deviations from planarity.

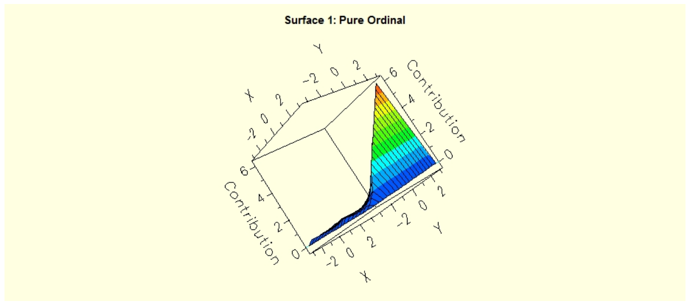
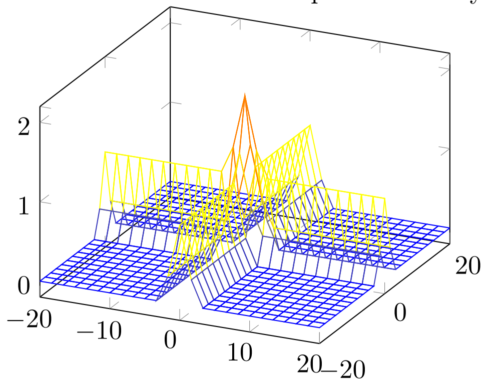


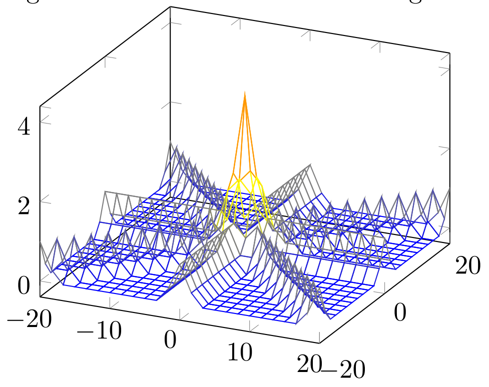
FIGURE 4. MARS on max(0,x+y)

We now turn to the hinged hyperplane (HHP) method of Leo Breiman. The basis functions in his technique consist of any two affine (non-homogeneous linear) functions combined by a maximum or minimum operator. To approximate any continuous function on a compact domain, such functions are added together. There is no need to subtract such functions or multiply them by constants because the same effect can be achieved by multiplying the functions in a hinge by some real constant and/or changing a max to a min. Breiman has a proof that a sum of hinged hyperplanes can approximate any continuous function on a compact domain uniformly to any desired accuracy.

Clearly HHP has no trouble to approximate $max(0, x + y)$. Breiman has proved that HHP can also approximate continuous functions like MARS. But it does so differently. We ask: how can HHP produce something like a bump? The difference with MARS is that it can't form products. So if we try to form a bump like MARS, the result is not local. We emphasize this by expanding the domain



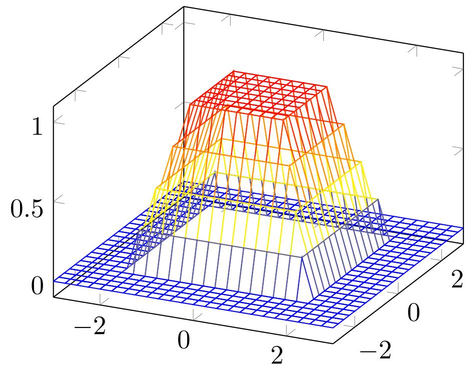
To show how HHP can get closer to just a bump, we rotate the above graph by 45 degrees and add it to the above to get:



This emphasizes the bump in the middle and reduces the size of the wings going out to the edges of the domain. Obtaining accuracy involves using using a huge number of hinges.

In summary, MARS and HHP, although universal approximators, are very inefficient at fitting certain functions. It is interesting that all one has to do is combine the ideas used in MARS and HHP to get a method that can fit functions in a very efficient manner. MARS and HHP both start with affine functions and take maxima and minima of these, but then they combine these hinges using addition. All that ALNs do is change the method of combining affine functions to using the maximum and minimum operators. ALNs are also smoothed by polynomials like MARS.

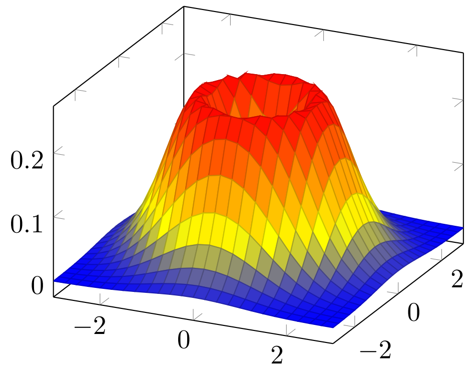
For example, ALNs can take the minimum of the two functions that were either multiplied (MARS) or added (HHP) to get a bump something like this.



That also makes clear that ALNs can approximate any continuous function on a bounded, closed domain universally to any degree of accuracy.

In what follows, we will show how more complicated functions can be approximated by MARS, HHP and ALN methods. For example, we call this function the "Devil's Tower" with equation $z = -exp(-(x^2 + y^2)) + exp(-0.5 * (x^2 + y^2))$.

Devil's Tower from the formula



REFERENCES

Allenby, G. (2004), A Choice Model for Packaged Goods: Dealing With Discrete Quantities and Quantity Discounts, Marketing Science.

Attoh-Okine, N. (2000), Flexible Pavement Roughness Prediction Using Adaptive Logic Networks, Journal of Smart Engineering System Design, 2, 257-271.

Attoh-Okine, N., and Roddis, W. (2004), eds., Computational Intelligence: From Theory to Practice, American Society of Civil Engineers.

Attoh-Okine, N, and Cogger, K. (2009), "Multivariate Adaptive Regression (MARS) and Hinged Hyperplanes (HHP) for Doweled Pavement Performance Modeling," Construction and Building Materials.

Bacon, D., and Watts, D. (1971), Estimating the transition between two intersecting straight lines, Biometrika, 58, 525-534.

Bollerslev, T. (1986), Generalized Autoregressive Conditional Heteroscedasticity, Journal of Econometrics, 31, 307-327.

Box, G.E.P., and Jenkins, G.M. (1976), Time Series Analysis: Forecasting and Control, revised edition, Holden-Day, San Francisco.

Brach, B., and Wang, J. (1995), Risk Arbitrage Performance for Stock Swap Offers with Collars, Presentation, Financial Management Association.

Bracker, K., Cogger, K., and Fanning, K., (2001), Currency Futures Trading Strategies: An Adaptive Logic Network Approach, The New Review of Applied Expert Systems, 7, 97-108.

Breiman L. (1993), Hinging Hyperplanes for Regression, Classification, and Function Approximation, IEEE Transactions On Information Theory, 999-1013.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984), Classification and Regression Trees, Wadsworth.

Charnes, A., Cooper, W., and Rhodes, E. (1978), Measuring the efficiency of decision making units, European Journal of Operational Research, 2, 429-444.

Cogger, K., Koch, P., and Lander, D. (1997) , A Neural Network Approach to Forecasting Volatile International Equity Markets, Advances in Financial Economics, 3, 117-157.

Cogger, K. (2001), Structural Comparison of Nonlinear Approximants, With Applicatiions, International Joint Conference on Neural Networks, Proceedings, Washington, DC.

Cogger, K. (2007), "Rating Rater Improvement: A Method for Estimating Increased Effect Size and Reduction of Clinical Trial Costs," Journal of Clinical Psychopharmacology, 27(4): 418-420.

Cooper, S. (1998), Multiple Regimes in U.S. Output Fluctuations, Journal of Business & Economic Statistics, 16, 92-100.

Dantzig, G., and Infanger, G. (1993), Multi-Stage Stochastic Linear Programs for Portfolio Optimization, Annals of Operations Research.

Engle, R. (1982), Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation, Econometrica, 50, 987-1007.

Engle, R. (2002), New Frontiers for ARCH Models, Working and Presentation Paper, University of California, Davis.

(www.stern.nyu.edu/fin/workpapers/papers2002/pdf/wpa02037.pdf)

Englehardt, N., Feiger, A., Cogger, K., Sikich, D., DeBrota, D., Lipsitz, J., Kobak, K., Evans, K., and Potter, W. (2006), Rating the Raters: Assessing the Quality of Hamilton Rating Scale for Depression Clinical Interviews in Two Industry-sponsored Clinical Drug Trials, Journal of Clinical Psychopharmacology, 26, 71-74.

Fair, R., and Jaffee, D. (1973), Methods of Estimation for Markets in Disequilibrium, Econometrica, 41.

Fanning, K., and Cogger, K. (1994), A Comparative Analysis of Artificial Neural Networks Using Financial Distress Prediction, Journal of Intelligent Systems in Accounting, Finance, and Management, 3, 241-252.

Fanning, K., and Cogger, K. (1995), Detection of Management Fraud: A Neural Network Approach, Journal of Intelligent Systems in Accounting, Finance, and Management, 4, 113-126.

Fanning, K. and Cogger, K. (1998), Neural Network Detection of Management Fraud Using Public Financial Data, Journal of Intelligent Systems in Accounting, Finance, and Management, 7, 21-41.

Feder, P.I. (1975), On Asymptotic Distribution Theory in Segmented Regression Problems-Identified Case, Annals of Statistics, 3, 49-83.

Feder, P.I. (1975), The Log Likelihood Ratio in Segmented Regression, Annals of Statistics, 3, 84-97.

Friedberg, L.(2000) , The Labor Supply Effects of the Social Security Earnings Test, Rev. Econ. Statist.

Friedman, J.H. (1991), Multivariate Adaptive Regression Splines, The Annals of Statistics, Vol. 19, No. 1, 1-141.

Granger, C. and Terasvirta, T. (1993), Modeling Nonlinear Economic Relationships, Oxford, U.K.: Oxford University Press.

Hamilton, J. (1989), A New Approach to the Economic Analysis of Nonstationary Time Series Subject to Changes in Regime, Econometrica, 57, 357-384.

Hansen, B. (1997), Inference in TAR Models, Studies in Nonlinear Dynamics and Econometrics, Vol.2 (1), 1-14.

Harvey, C. (2001), The Specification of Conditional Expectations, Journal of Empirical Finance.

Heim, B., and Meyer, B (2003)., Structural Labor Supply Models When Budget Constraints Are Nonlinear, Working Paper, Yale University.

Hinkley, D. (1969), Inference About the Intersection in Two-Phase Regression, Biometrika, 56, 495-504.

Hinkley, D. (1971), Inference in Two-Phase Regression, J.Amer.Statist.Assoc., 66,736-743.

Hirschberg, J., and Lye, J. (2001), Clustering in a DEA Using Bootstrapped Efficiency Scores, Working Paper, University of Melbourne.

Hudson D.J. (1966), Fitting Segmented Curves Whose Join Points Have to be Estimated, Journal of the American Statistical Association, 1097-1129.

Konno, H., and Yamazazaki, H. (1991), Mean-Absolute Deviation Portfolio Optimization Model and its Application to the Tokyo Stock Market, Management Science.

Lee, J., and Brown, M. (1985), Coupon Redemption and the Demand for Frozen Concentrated Orange juice-A Switching Regression Analysis, American Journal of Agricultural Economics, 67 647-653.

Luce, R., and Tukey, J. (1964),Simultaneous Conjoint Measurement: A New Type of Fundamental Measurement, Journal of Mathematical Psychology, 1, 1-27.

Mathiesen, H. (2006), Empirical Studies on Ownership Structure and Performance,

`www.encycogov.com/A5OwnershipStructures/OwPerfStudies/Table_Ow_HIJ.asp`

.

McGee, V., and Carleton, W. (1970), Piecewise Regression, Journal of the American Statistical Association, 65, 1109-1124.

Medeiros, M., Veiga, A., and Resenda, M. (2002), A Combinatorial Approach to Piecewise Linear Time Series Analysis, J. Computational and Graphical Statistics.

Merz, M. (1999), Time Series Evidence of Unemployment Flows: The Sample Period Matters, Journal of Business & Economic Statistics, 17, 324-334.

Moffit, R. (1986), Estimating Consumer Demand Functions When Budget Constraints Are Piecewise Linear, J. Bus. Econ. Stat.

Morck, R., Shleifer, A., and Vishny, R. (1988), Management Ownership and Market Valuation, an Empirical Analysis, Journal of Financial Economics, 20, 293-315.

Meullenet, J., and Xiong, R. (2004), Application of MARS to the Preference Mapping of Cheese Sticks, J. Food Science.

Proietti, T. (1998), Characterizing Asymmetries in Business Cycles Using Smooth-Transition Structural Time-Series Models, Studies in Nonlinear Dynamics and Econometrics, 3(3), 141-156.

Quandt, R. (1958), The Estimation of the Parameters of a Linear Regression System Obeying Two Separate Regimes, Journal of the American Statistical Association, 53, 873-880.

Quandt, R. (1960), Tests of the Hypothesis that a Linear Regression System Obeys Two Separate Regimes, Journal of the American Statistical Association, 55, 324-330.

Rao, A., Miller, D., Rose, K., and Gersho, A. (1999), A Deterministic Annealing Approach for Parsimonious Design of Piecewise Regression Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.21, No.2, 159-173.

Reiss, P., and White, M. (2005), Household Electricity Demand, Revisited, Rev. Econ. Studies.

Ritchken, P. (1985), On Option Pricing Bounds, Journal of Finance.

Seiford, L. (1995), A Data Envelope Analysis Bibliography (1978-1992), in A. Charnes, W. Cooper, A. Levin, and L. Seiford (eds.), Data Envelope Analysis: Theory, Methodology and Applications, Boston, Kluwer Academic Publishers.

Shaban, S. (1980), Change Point Problem and Two-Phase Regression: An Annotated Bibliography, International Statistical Review, 48, 83-93.

Suits, D. (1955), Econometric Model of the Water Melon Market, Journal of Farm Economics, 37, 237-251.

Tong, H., and Wu, Z. (1982), Multi-Step-Ahead Forecasting of Cyclical Data by Threshold Autoregression, in Time Series Analysis: Theory and Practice, O.D. Andersen, ed., New York:North Holland, 733-753.

Tong, H. (1983), Threshold Models in Non-linear Time Series Analysis New York: Springer-Verlag.

Tong, H. (1990), Non-Linear Time Series: A Dynamical System Approach, New York: Oxford University Press.

Tsay, R. (1998), Testing and Modeling Multivariate Threshold Models, Journal of the American Statistical Association, Vol.93, No.443, 1188-1202.

Velilla, S.(1998) , Assessing the Number of Linear Component in a General Regression Problem, Journal of the American Statistical Association, 93, 1088-1098.

Zhan (1999), The Econometrics of Piecewise Linear Budget Constraints: An Application to Housing Demand in the Presence of Capital Gains Taxation, PhD

ALN Bibliography (post 1994)

Kostov, A., Strange K., Stein, R.B., and Hoffer A.J., (1995), Adaptive Logic Networks in EMG-prediction from Sensory Nerve Signals Recorded in the Cats Forelimb During Walking, Physiology Canada, Vol. 26, No. 2, pp. 104.

M J Polak, SH Zhou, P M Rautaharju, W. W. Armstrong, B R Chaitman Adaptive logic network compared with backpropagation network in automated detection of ischemia from resting ECG, Proc. Computers in Cardiology Conf., Vienna, Austria, Sept. 10 - 13, 1995, pp 217 - 220.

W. W. Armstrong, C. Chu, and M. Thomas, Using adaptive logic networks to predict machine failure, Proc. World Congress on Neural Networks (WCNN'95) Washington DC, July 1995.

Aleksandar Kostov, Brian J. Andrews, Dejan B. Popovic, Richard B. Stein, William W. Armstrong, Machine Learning in Control of Functional Electrical Stimulation Systems for Locomotion, IEEE Trans. Biomed. Eng. vol. 42 no. 6, 1995, pp. 541 - 551.

Richard B. Stein, Aleksandar Kostov, Dejan Popovic, William W. Armstrong, and Monroe Thomas, Functional Electrical Stimulation aided locomotion controlled in real time by artificial neural networks, Can. J. Physiol. Pharmacol. 73:A29, 1995 (Abstract)

A. Kostov, R. B. Stein, W. W. Armstrong, M. Thomas, D. Popovic, Integrated control system for FES-assisted locomotion after spinal cord injury, Proc. IEEE Engineering in Medicine and Biology Society, 17th Ann. Conf., Montreal, September 20-23 1995 (CD-ROM).

W W Armstrong, Monroe M Thomas, Adaptive Logic Networks, Section C 1.8 in Handbook of Neural Computation, Emile Fiesler and Russell Beale eds., Oxford University Press, 1996, ISBN 0-7503-0312-3 (looseleaf)

W W Armstrong, Monroe M Thomas, Neural net control of an active suspension system, Section G2.1 Handbook of Neural Computation (ibid)

Kostov, A., Armstrong, W.W., Thomas M., and Stein, R.B., Case Study - Adaptive Logic Networks in Rehabilitation of Persons with Incomplete Spinal Cord Injury, Section G.5.1, Handbook of Neural Computation, ibid.

D.O.Gorodnichy, W.W.Armstrong, X. Li, Adaptive Logic Networks for Facial Feature Detection, Lecture Notes in Computer Science, Vol 1311, Springer Verlag, 1997, pp. 332-339. Proc. of 9th Intern. Conf. on Image Analysis and Processing ICIAP'97, Florence, Italy, Sept. 1997.

M. J. Polak, S. H. Zhou, P. M. Rautaharju, W. W. Armstrong, B. R. Chaitman, Using automated analysis of resting twelve-lead ECG to identify patients at risk of developing transient myocardial ischaemia – an application of an adaptive logic network, Physiological Measurement 18, 1997 pp.317 - 325.

W. W. Armstrong, Reinforcement learning applied to simulated basketball balancing, Int'l. ICSC-IFAC Symposium on Neural Computation, NC'98, Vienna, Sept 23 - 25, 1998, CD-ROM ISBN 3-906454-14-2

S. Ramaswamy, T. Ono-Tesfaye, W. W. Armstrong and P. Gburzynski, Equivalent Bandwidth Characterization for Real-time CAC in ATM Networks, Journal of High Speed Networks, 1998, pp 1-25.

William W. Armstrong and Darwin Li, A new technique for reinforcment learning in control, 1998 IEEE Conf. on Systems, Man and Cybernetics, La Jolla CA, Oct. 11 -14, 1998, ISBN 0-7803-4781-1, IEEE cat. no. 98CH36218.

D.O. Gorodnichy, W.W. Armstrong, Single Camera Stereo for Mobile Robots, Proc. Vision Interface (VI'99), Trois Rivieres, Canada, May 18-21, 1999, pp. 528-535.

D.O. Gorodnichy, W.W. Armstrong, A Parametrical Alternative for Grids in Occupancy Based World Modeling, Proc. Quality Control by Artificial Vision Conference (QCAV'99), Trois Rivieres, Canada, May 18-21, 1999, pp 125 - 132.

W.W. Armstrong, B. Coghlan, D.O. Gorodnichy, Reinforcement learning for autonomous robot navigation, Proc. Int'l Joint Conf. on Neural Networks (IJCNN'99), Washington DC, July 21-23, 1999

G. Qu, J. J. R. Feddes, W. Armstrong, J. Leonard, R. Coleman, Combining and Electronic Nose with an Artificial Neural Network to Measure Odour Concentration, ASAE/CSAE-SCGR Annual Int'l Meeting, July 18-21, 1999, Toronto (not refereed).

G.Qu, J.J.R.Feddes, W.W.Armstrong, R.N.Coleman and J.J.Leonard, Measuring odor concentration with an electronic nose, Proc. Second International Conference on Air Pollution from Agricultural Operations, October 9-11, 2000, Des Moines, Iowa. pp 188-195. Library of congress card number (LCCN)00-134838. International standard book number (ISBN)1-892769-12-3. ASAE Publication 701P003.

W. W. Armstrong, D. O. Gorodnichy, Breaking Hyperplanes to fit Data with Applications to 3D World Modeling and Oil Sand Data Analysis, Proc. ICSC Symposium on Neural Computation, NC 2000, Berlin, May 23 - 26, 2000, CD-ROM publ. by ICSC Academic Press, Int'l Comp. Sci. Conv., Canada/Switzerland ISBN 3-906454-21-5

Dmitry O. Gorodnichy, W. W. Armstrong, Neurocomputational Approach for Modeling Large Scale Environments from Range Data, ibid.

W. W. Armstrong, High speed networks that preserve continuity and accuracy, Int'l Joint Conf. on Neural Networks, Washington DC, July 14 - 19, 2001 - Special session on morphological networks, CD-ROM.

W. W. Armstrong, 3624 - 108 Street NW, Edmonton, AB, Canada
E-mail address: `wwarmstrong@shaw.ca`