Neural Network Hedonic Pricing Models in Mass Real Estate Appraisal

Authors Steven Peterson and Albert B. Flanagan

Abstract

Using a large sample of 46,467 residential properties spanning 1999–2005, we demonstrate using matched pairs that, relative to linear hedonic pricing models, artificial neural networks (ANN) generate significantly lower dollar pricing errors, have greater pricing precision out-of-sample, and extrapolate better from more volatile pricing environments. While a single layer ANN is functionally equivalent to OLS, multiple layered ANNs are capable of modeling complex nonlinearities. Moreover, because parameter estimation in ANN does not depend on the rank of the regressor matrix, ANN is better suited to hedonic models that typically utilize large numbers of dummy variables.

Relative illiquidity, low turnover, and irregularly timed (or absent) cash flows confound the application of standard asset pricing models to real estate. In general, non-exchange traded assets such as private residential real estate are characterized by a lack of fundamentals and, thus, valuation is less a function of discounted present value than one of finding recently traded assets of comparable value.

It is the absence of asset fundamentals that gives rise to hedonic valuation models that extrapolate from means in large samples. These models essentially predict value by projecting a sample of known market values on their respective property characteristics (such as heated area, square footage, age, acreage), using the estimated parameters in conjunction with a vector of characteristics for a property of unknown value to imply a price. Large-scale implementation of linear hedonic models can be found, for instance, in automated valuation systems adopted in the mortgage finance industry (Rossini, Kershaw, and Kooymans, 1992, 1993; Detweiler and Radigan, 1996; and Baen and Guttery, 1997). Because these models are easy to specify, estimate, and extrapolate, they tend to be popular to end-users.

Nevertheless, usefulness depends on how well models minimize pricing errors. Pricing errors are responsible in part for denial of credit to otherwise credit worthy parties resulting in type I error and goodwill loss, as well as extension of credit to parties with underestimated risk—a form of type II error (Shiller and Weiss, 1999). Hedonic models are exposed to pricing errors simply because they extrapolate means from large samples and, as such, will always be exposed to

sampling error. Specification error is also unavoidable in ad hoc specifications and, to the extent that value does not map linearly onto property characteristics, so too are errors due to neglected nonlinearities.

To the extent that nonlinear models nest linear forms, then nonlinear models would be the preferred choice. However, the exact nonlinear form is neither apparent nor are there necessarily practical steps one could take to find the correct form. Artificial neural networks (ANN) do, however, provide a practical alternative to conventional least squares forms (including nonlinear least squares) that is easily implementable and which efficiently models nonlinearities in the underlying relationships (including the parameters).

We argue that neural networks are more robust to model misspecification and especially to various peculiarities in how various explanatory variables are measured. Hedonic models rely heavily on property attributes and therefore many of the explanatory variables are categoricals or counts. Categoricals, such as location, zoning, or type of construction material, have no ordinal rankings and therefore do not belong in the regression function except in the form of dummy variables. The regressor matrix is often dominated by dummies and, as we show below, this increases the likelihood of rank failure. Linear models deal with this problem by aggregating cases within categories to produce fewer dummies but at a cost of discarding useful information that helps discriminate between properties. Still, other variables, such as number of baths or story height, while ordinal, are quite limited in their ranges. Nevertheless, these are often incorporated (perhaps with their squares as proxies for nonlinearities) as if they were continuously measured regressors.[1] In fact, the model studied in this paper is restricted in the number and type of usable explanatory variables because OLS estimates were impossible to estimate due to rank failure in the matrix of regressors—a problem not inherent in feed forward networks, which do not require inverting the matrix of inputs. We will comment further on this point below.[2]

Despite this, mass appraisal and automated valuation systems tend to rely on linear models partly due to convenience and to some degree because the costs of pricing errors are not fully understood. In this paper, we show, using a sample of 46,467 residential property sales from 1999 to 2005, that convenience can be expensive. We illustrate using matched pair *t*-tests of property valuations from ANN against linear hedonic pricing models, that the latter generate statistically significant greater pricing error, that the magnitude of this error has become larger over time, and that ANN has greater relative pricing precision in-sample, as well as out-of-sample. Finally, we measure the relative dollar savings as ANN minimizes pricing error as a mass appraisal tool.

The paper proceeds as follows. Section two contains a brief review of the literature and discussion of the underlying models. We note beforehand that while there is no consensus in the literature, many of the criticisms of neural networks in real estate valuation are based on small samples. One of our contributions to this literature is the large size of our database and the introduction of robust statistical

tests as both our training and hold-out samples number in the thousands for each year of the study. Section three discusses the model specifications and the test methodology. Section four presents the results and section five offers some concluding remarks and suggestions for further research.

Artificially Intelligent Hedonic Pricing Models

There are conflicting views on the general relative performance of multiple regression based hedonic pricing models and neural networks. Studies by Tsukuda and Baba (1990), Do and Grudnitski (1992), Tay and Ho (1992), and Huang, Dorsey, and Boose (1994) all found neural networks to be superior to multiple regression. Allen and Zumalt (1994) and Worzala, Lenk, and Silva (1995) suggested otherwise. In a more recent study, Guan, Zuarda, and Levitan (2008) combine fuzzy set theory in neural network architecture to assess property values extending the seminal work of Bagnoli, Smith, and Halbert (1998), who originally applied fuzzy logic to real estate evaluation.

Nguyen and Cripps (2001) compared neural networks to multiple regression models based on a dataset of single family houses and found that neural networks outperformed multiple regression models when the size of the dataset is large. In their study, neural network models tended to overcome functional form misspecification as the sample size increased and while multiple regression performance was relatively independent of sample size, neural network performance improved. Our analysis supports these conclusions.

Other studies find significant nonlinearities between home value and age (Grether and Mieszkowski, 1974; and Do and Grudnitski, 1993) and home value and square footage (Goodman and Thibodeau, 1995). Although the presence of nonlinear mappings from factors such as age, size, and distance to home value, are generally accepted, the cost of excluding them from the valuation model remains a topic of debate. Our findings suggest that the cost is significant; pricing errors in our linear models are significantly greater than those for our neural networks.

Worzala, Lenk, and Silva (1995) compared two neural networks to multiple regression models in the application of real estate appraisal. Their study, which was based on a small set of transactions within one town,[3] concluded that neural networks were not superior to multiple regressions in residential real estate appraisal and warned appraisers who wish to use neural networks to do so with caution citing inconsistent results between software packages, between runs of the same software package, and long run-times. It is indeed true that neural networks can be over-trained resulting in good training runs but poor out-of-sample performance, that there is some sensitivity to outliers in training, and that results may be inconsistent in small samples. Results may also appear inconsistent for the simple reason that network weights are often randomly initialized; thus, gradient descent algorithms produce different solutions for the network's weight vectors simply because they begin iterating from different points on the loss

surface. Ensemble averaging (Haykin, 2003) can be used effectively in dealing with this issue. On the other hand, software packages can produce different results simply because they utilize different learning algorithms and performance criteria for training. Many programming languages (we use Matlab) allow the user complete control of network design, training criteria, and simulation design and long run times have been eliminated with advances in processing power. As such, these criticisms, though not without merit, are hardly binding.

To the contrary, Guan, Zurada, and Levitan (2008) argue that neural networks better replicate agents' heuristic thought processes, as well as the imprecision in their decision calculus. Given the volume of research devoted to quasi-rational thought processes, such as various mental accounting rules (Thaler, 1985) and representative heuristics (Kahneman and Tversky, 1972), then neural-based models that incorporate fuzzy rules of logic are exciting developments in the field of property value assessment.

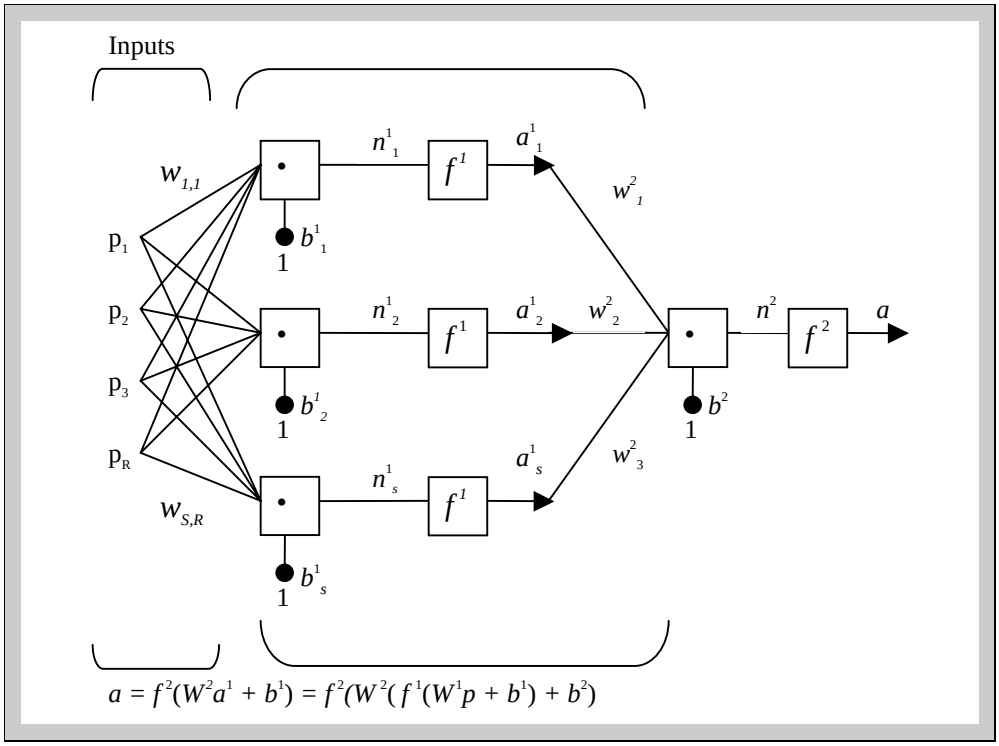
Critics of the neural networks also cite the relative ease of interpretation of hedonic multiple regression models; in particular, partial differentiation of linear models easily isolates each explanatory variable's contribution to value. Although differentiation of neural networks is more difficult given variable interdependencies, it is relatively straightforward to uncover individual variable attributions (Garson, 1991; and Intrator and Intrator, 2001). Thus, while the contribution of, say, square footage, to home value in a neural network cannot be reduced to a single beta, it can nevertheless be assessed by other means (e.g., simulation methods). At any rate, mass appraisal has relied primarily on a multiple regression framework despite the problems associated with nonlinearities, non-normality of inputs, and multicollinearity.

The ANN design, depicted in Exhibit 1, is a standard feed forward network with a single, hidden layer, trained by propagating errors back through the network, adjusting nodal weights toward the goal of minimizing the sum of squared errors between the network output and the target values (dependent variable). The $R \times 1$ input vector $\mathbf{p}$ is a vector of attributes, say, for a single property, whose value in dollars is given by the scalar target, $\mathbf{t}$. These inputs are fed into the network and weighted by $\mathbf{W}$, an $S \times R$ matrix, where S is the number of nodes in the network layer. The $S \times 1$ vector $\mathbf{b}$ are the network layer biases—the analogue to the intercept in linear least squares. Each nodal output is the sum of the weighted input vector. The transformation function $\mathbf{f}$ compresses these nodal outputs, feeding the transformed values forward to the output layer as $\mathbf{a}^1$. Thus, the first layer output is:

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}). \quad (1)$$

The transformation function is not arbitrary. We use an arctangent that transforms the weighted inputs into the bounded interval $\{-1 \leq \mathbf{a}^1 \leq 1\}$. These values represent the signal strength that each node in layer one feeds to the output layer.

Exhibit 1 | The Structure of a Neural Network with *s* Hidden Nodes



Hidden nodes serve as ''feature detectors'' (Haykin, 2003); the signal sent to the output layer to be compared to the target (dependent variable) is a weighting of the hidden nodal output, themselves weighted functions of the inputs. Because hidden nodes weight inputs independently of one another, they present contrasting representations of the relationships between inputs and targets.

The hidden layer output $\mathbf{a}^1$ is then weighted and transformed into the final output to compare to the target values. Since our target is a dollar value, then the transformation function in the output layer is a simple, unbounded, linear transform, i.e., it is simply the weighted sum of $\mathbf{a}^1$:

$$\mathbf{a} = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1) = \mathbf{f}^2(\mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b})),$$

where $\mathbf{f}^2$ is a simple sum. (2)

The network weights are trained on repeated property vectors, iterating on the weights to minimize a sum of squared error loss function:

$$F = {}_{w}\min \sum (t - a)^2. \tag{3}$$

Training feed forward networks is by gradient descent (e.g., Newton's algorithm) in which the learning process is revealed in the evolution of weights via:

$$w^s_{i,j}(k + 1) = w^s_{i,j}(k) - \alpha \frac{\partial F}{\partial w^s_{k,j}}, \tag{4}$$

where $w_{i,j}$ is the weight on the jth input in the ith node in layer s, α is the learning rate, and the partial derivative is the gradient. Errors are propagated backwards through the network so that weight adjustments begin in the first layer and then proceed forward to the output layer weights. Though standard textbook fare (see any one of the texts cited below), we include a brief outline of the process in the Appendix. We present below results from a three node single hidden layer design with R = 9 inputs and S = 3 nodes in a single hidden layer.

Experimental Design and Tests

We study a sample of 46,467 properties transacted over the period 1999–2005. These observations were taken from a database of over 180,000 observations of residential sales in Wake County, NC, part of the Raleigh-Cary Metropolitan Statistical Area. The data are part of the real estate master file, which is county government data used in property valuation and taxation on real estate. The database includes commercial and residential sales data in 20 townships, over 80 specific property types and over 18 property characteristics.

Our primary objective is to compare appraisal performance for linear hedonic models relative to ANNs. To that end, we test for statistical significance in relative pricing errors using matched paired *t*-tests. Both models share the same inputs—age, number of units, lot size (acreage), number of stories, heated area, number of baths, and a dummy variable indicating exterior composition (wood, masonry, vinyl or aluminum siding). The dependent variable is the observed sale price.[4] Summary statistics for these variables are presented in Exhibit 2.

Sampling a proportion p of N homes generates a training group of size pN and a hold-out of size $(1\text{-}p)N$. The experimental design consists of drawing 100 of these randomly selected size pN training samples of homes in each year reserving the remaining $(1\text{-}p)N$ home sales for that year as a hold-out sample. Each training sample was used to train the neural network and, separately, to estimate the parameters of the linear model (using OLS).

Exhibit 2 | Summary Statistics for Dependent Variable and Model Inputs

Variable	Mean	Std. Dev.	Min	Max
Sale Price	208,581.10	124,350.20	5	2,650,000
Age	14.57	15.57	0	192
No. of Units	1.01	0.27	0	14
Lot size, acres	0.42	1.04	0.03	96.5
No. of Stories	2.74	2.06	1	9
Heated Area, Sq. ft.	2,018.52	854.50	0	19,935
Baths	4.46	2.18	1	10
Exterior Composition[a]			1	3
Sale Price	208,581.10	124,350.20	5	2,650,000
Age	14.57	15.57	0	192
No. of Units	1.01	0.27	0	14
Lot size, acres	0.42	1.04	0.03	96.5
No. of Stories	2.74	2.06	1	9
Heated Area, Sq. ft.	2,018.52	854.50	0	19,935
Baths	4.46	2.18	1	10
Exterior Composition[a]			1	3

Notes: The number of observations is 46,467.
[a]Exterior composition: 1 = wood, 2 = masonry, and 3 = aluminum/vinyl.

Absolute pricing errors are computed for the training sample (these are equivalent to regression's ''in-sample'' results) and the estimated models are used to forecast the prices in the hold-out sample. In this fashion, we generate a random sample of paired pricing errors at the property level. The pricing error differential is the statistic of interest.

More specifically, let e_i represent the *absolute* pricing error on property i and denote the training and hold-out samples by superscripts T and HO respectively. Interest centers on the size of the pricing error differential between the linear and neural network pricing models. For the target and hold-out samples, respectively, we are interested in the distribution of the statistics:

$$\sum_{i=1}^{pN} (e_i^{T,OLS} - e_i^{T,NN})/pN; \quad p = \{0.1, 0.25, 0.5, 0.75\} \tag{5}$$

$$\sum_{i=1}^{(1-p)N} (e_i^{HO,OLS} - e_i^{HO,NN})/(1-p)N; \quad p = \{0.1, 0.25, 0.5, 0.75\} \tag{6}$$

Equation (5) summarizes the in-sample results for each training sample while (6) pertains to the hold-out samples. The statistic is the mean absolute pricing error differential—a positive value favors ANN. We analyze the distribution of this statistic in 100 randomly drawn samples. The null hypothesis we test is that the mean mispricing error is zero.[5] We report the grand mean dollar mispricing in Exhibit 3 and observe, by year, the magnitude of relative mispricing. Statistical tests and additional fit statistics are presented in Exhibit 4.

That is the basic experiment. This design is extended to accommodate various hold-out and training sample sizes. Specifically, we replicate the experiment for randomly selected training samples of sizes *p* equal to 10%, 25%, 50%, and 75% of the population of properties in each year. (The hold-out samples were the complementary samples.) This extension permits us to locate possible bias in model performance especially as it relates to the relative ability of each model to extrapolate out of sample from various training sample sizes.

We also test in-sample and hold-out-sample pricing performance and present differences in root mean squared errors (RMSE), mean absolute pricing errors (MAPE), and the linear model's R-squared statistics.

Results

The pricing error differential is the difference between the linear and neural network absolute forecast errors on a property-by-property basis. Positive differentials favor the neural network. The general pattern in Exhibit 3, Panels A and B, clearly favors the neural network regardless of sample size, and in general, this pattern holds both in-sample and for the hold-out-sample. The size of the pricing error tends to increase over time. To place these results in the proper context, Panel B summarizes mean home values and pricing errors as a percentage of mean value, both in-sample and out-of-sample, as well as for the various holdout sample sizes. Clearly, OLS performance is inversely related to the size of the training sample and suffers significantly as it is required to generalize to ever larger hold-out samples. Moreover, pricing errors are larger more recently, which may suggest either, or both, increasing price volatility[6] of a shift in the relationship between house price and explanatory variables that are not captured by the linear model; note in Exhibit 4 that R-squared values are virtually constant over time (i.e., the linear model fails to capture the increased volatility in home prices).

It is also interesting that the neural network extrapolates better from larger training sets, e.g., the performance of the linear model is somewhat flat as indicated by

Exhibit 3 | In-Sample and Out-of Sample Pricing Errors

	Pricing Error			
Year	10%	25%	50%	75%
Panel A: Pricing Errors by Year and Size of the Holdout Sample[a]				
In-Sample				
1999	$2,958.00	$2,046.17	$1,912.90	$1,730.97
2000	$4,507.70	$3,263.46	$2,790.26	$2,470.16
2001	$4,571.80	$2,654.38	$2,279.28	$2,091.61
2002	$4,028.00	$3,006.66	$2,683.50	$2,612.21
2003	$3,057.40	$2,987.70	$2,696.22	$2,846.33
2004	$4,873.80	$4,454.90	$4,452.69	$4,364.11
2005	$4,515.20	$3,728.38	$3,692.32	$3,387.69
Average	$4,073.13	$3,163.09	$2,929.60	$2,786.15
Out-of-Sample				
1999	$279.81	$881.35	$1,366.39	$1,353.00
2000	−$949.42	$1,158.87	$1,404.56	$958.95
2001	−$889.23	$576.94	$1,015.22	$1,300.24
2002	$344.00	$1,666.34	$2,012.87	$2,079.52
2003	$1,454.70	$2,211.40	$2,330.99	$2,412.35
2004	$3,511.40	$3,886.46	$3,904.37	$3,712.98
2005	$2,248.30	$2,735.82	$3,242.15	$3,062.32
Average	$857.08	$1,873.88	$2,182.36	$2,125.62

Panel B: Relative Percentage Pricing Errors[b]

		Pricing Error			
Year	Mean Value	10%	25%	50%	75%
In-Sample					
1999	$181,849.00	1.63%	1.13%	1.05%	0.95%
2000	$187,061.00	2.41%	1.74%	1.49%	1.32%
2001	$189,689.00	2.41%	1.40%	1.20%	1.10%
2002	$194,597.00	2.07%	1.55%	1.38%	1.34%
2003	$202,276.00	1.51%	1.48%	1.33%	1.41%
2004	$220,165.00	2.21%	2.02%	2.02%	1.98%
2005	$228,385.00	1.98%	1.63%	1.62%	1.48%
Average	$200,574.57	2.03%	1.58%	1.46%	1.39%
Out-of-Sample					
1999	$181,849.00	0.15%	0.48%	0.75%	0.74%
2000	$187,061.00	−0.51%	0.62%	0.75%	0.51%
2001	$189,689.00	−0.47%	0.30%	0.54%	0.69%
2002	$194,597.00	0.18%	0.86%	1.03%	1.07%
2003	$202,276.00	0.72%	1.09%	1.15%	1.19%
2004	$220,165.00	1.59%	1.77%	1.77%	1.69%
2005	$228,385.00	0.98%	1.20%	1.42%	1.34%
Average	$200,574.57	0.43%	0.93%	1.09%	1.06%

Notes: Feedforward ANN with three hidden nodes.
[a]Average of matched pairs OLS pricing error—ANN pricing error
[b]Pricing error relative to ANN as percent of mean home value.

Exhibit 4 | RMSE and MAPE by Year and Size of Holdout Sample[a]

Year	*N*	*t*-Stat	MAPE(ANN)	MAPE(OLS)	RMSE In-Sample Difference	RMSE Holdout Difference	R^2(OLS)	RESET[b]
Panel A: Holdout Sample 10%								
1999	444	0.80	0.220	0.241	−5,734.9	2,246	0.80	11.82
2000	368	−0.95	0.226	0.251	−9,366.1	−2,760.9	0.74	12.55
2001	379	−1.66	0.195	0.217	−1,2254	4,760.9	0.72	14.08
2002	460	0.99	0.201	0.220	−6,908	3,174.7	0.77	12.42
2003	804	5.11	0.175	0.193	−5,809.6	−1,301.7	0.75	13.95
2004	1,152	10.93	0.209	0.235	−7,645.6	−5,894.1	0.73	18.26
2005	959	6.91	0.260	0.285	−7,162.9	633.6	0.74	14.30
Average	652	3.16	0.210	0.230	−7,840.2	122.7	0.75	13.91
Panel B: Holdout Sample 25%								
1999	1,109	2.64	0.229	0.245	−3,364.0	343.7	0.79	14.1
2000	920	1.73	0.237	0.255	−7,132.0	−6,239.0	0.72	17.4
2001	947	1.30	0.201	0.213	−6,230.7	459.6	0.70	15.3
2002	1,151	4.60	0.210	0.225	−4,003.7	179.1	0.75	14.2
2003	2,009	7.90	0.176	0.194	−4,581.6	−2,469.5	0.75	18.6
2004	2,880	12.87	0.209	0.235	−5,759.7	−5,301.0	0.73	23.6
2005	2,398	8.40	0.251	0.267	−4,178.0	−745.9	0.74	16.9
Average	1,631	5.63	0.220	0.230	−5,035.7	−1,967.6	0.74	17.2

Exhibit 4 | (continued)

RMSE and MAPE by Year and Size of Holdout Sample[a]

Year	*N*	*t*-Stat	MAPE(ANN)	MAPE(OLS)	RMSE In-Sample Difference	RMSE Holdout Difference	R^2(OLS)	RESET[b]
Panel C: Holdout Sample 50%								
1999	2,218	3.69	0.226	0.240	−2,457.2	−572.1	0.78	17.5
2000	1,841	1.99	0.244	0.258	−7,365.1	−6,804.9	0.70	23.5
2001	1,894	2.16	0.199	0.211	−4,338.4	−1,110.9	0.70	17.6
2002	2,301	4.88	0.208	0.222	−2,881.8	−919.1	0.75	16.7
2003	4,018	7.36	0.179	0.195	−3,560.4	−2,739.4	0.74	23.8
2004	5,760	11.48	0.211	0.235	−5,196.3	−4,503.0	0.72	31.5
2005	4,797	8.56	0.258	0.274	−3,624.4	−1,625.0	0.74	20.9
Average	3,261	5.73	0.220	0.230	−4,203.4	−2,610.6	0.73	21.6
Panel D: Holdout Sample 75%								
1999	3,327	2.63	0.226	0.238	−2,018.7	−847.4	0.78	17.9
2000	2,761	1.26	0.249	0.262	−7,493.4	−5,778.2	0.70	26.1
2001	2,841	2.09	0.201	0.212	−3,535.7	−1,646.7	0.69	19.0
2002	3,452	3.71	0.211	0.225	−2,429.6	−1,388.4	0.75	19.8
2003	6,027	5.36	0.178	0.195	−3,689.1	−2,927.5	0.74	28.3
2004	8,640	7.71	0.210	0.234	−4,792.2	−4,569.4	0.72	37.6
2005	7,195	5.99	0.259	0.272	−2,898.5	−2,453.5	0.74	22.6
Average	4,892	4.11	0.220	0.230	−3,836.8	−2,801.6	0.73	24.5

Notes: RMSE Difference = ANN(RMSE) − OLS(RMSE).

[a]MAPE: See footnote 4.

[b]RESET test for neglected nonlinearities.

the R-squared statistics while the pricing error continues to increase with training sample size. In many cases, especially in more recent years, this error differential easily exceeds 1.5% of property value per year ($15 million on a $1 billion portfolio). That may seem small, but it is nevertheless statistically significant, and for mass appraisals (say, by mortgage lenders) this represents a considerable annual dead-weight loss potential, e.g., pricing errors lead to default losses, denial of credit, as well as LTV errors.

Additional statistical results on root mean squared error (RMSE) differential and mean absolute pricing errors (MAPE) are given in Exhibit 4.[7] Both statistics are error metrics expressed in dollar amounts. The story here reinforces that for the relative pricing error told above. Relative differences in RMSE, in particular, are on the same magnitude as pricing errors and get larger over time (absolutely and proportionately) and with the size of the training group. The same is true for MAPE, which for the linear model, tells the same story as R-squared, i.e., R-squared values in the neighborhood or 75% suggest mean absolute pricing errors in the 20%–25% range.

Proponents of ANN argue that a weakness of linear models is neglected nonlinearities. We use a RESET test in which the predictions of the ANN are included as an additional regressor in the OLS model. Thus, if the OLS specification is:

$$y = X\beta + \varepsilon, \tag{7}$$

and the predictions of the ANN are contained in the vector **m**, then a test of neglected nonlinearities is equivalent to a *t*-test on *HO*: $\varphi = 0$ in the regression:

$$\hat{\varepsilon} = X\beta + \varphi m + v. \tag{8}$$

Results from this test are included in Exhibit 4 as the average *t*-statistic over the 100 random samples from each year. The null of no neglected nonlinearities is easily rejected at any conventional level of significance. We also note that the mean size of this statistic rises with the sample size.

We have argued above that there are relevant differences in the manner in which ANN and OLS handle potential data problems, especially as these relate to rank failure in the matrix of regressors. Our point is that the presence of collinearity (or potential for rank failure) is a driving force in model specification. Even near-collinearity influences specification to the extent that standard errors are driven down, which often leads to respecification (we see few studies with insignificant regressors). When the rank condition does not fail, we still observe very high condition numbers for the covariance matrices of regressors (near zero

eigenvalues), which suggest instability in the OLS estimates. To illustrate, we constructed a set of dummies covering number of units (truncating these at five or more units), exterior composition (three dummies as noted above), story height (truncating at four or more stories), and number of baths (baths do not exceed ten in number). This data set had a total of 22 dummies but with rank equal to 19. There were therefore three redundant sources of information. While we could not estimate the model with OLS without further restrictions, we could with ANN because backpropagation does not invert the matrix of regressors.

Conclusion

We have shown that linear appraisal methods generate significant mispricing errors relative to a basic feed forward nonlinear artificial neural network. These results are robust; the sample of roughly 46,000 property sales spanning the seven-year period 1999–2005 produces ample degrees of freedom regarding our statistical tests and the randomization scheme for selecting hold-out and training groups reduces any effects due to sampling error. Our major conclusion is that linear hedonic valuation models produce avoidable valuation costs and that these costs are due primarily to nonlinearities in the relationships between property characteristics and value. And, while artificial neural networks may be one of several nonlinear methodologies, methods such as nonlinear least squares are impractical primarily because there is little guidance directing functional form.

Much of the data on property characteristics that hedonic models rely upon are discretely valued as either simple counts (number of baths or stories) or categorical (location code). In such instances, the matrix of explanatory variables consists primarily of a large number of dummy variables, often with failed rank condition. Thus, the covariance matrix of explanatory variables cannot be inverted (without first reducing the number of variables in the regression) or, if invertible, producing very imprecise estimates of the pricing model's coefficients. Backpropagation, on the other hand, splits the influence of redundant information across the nodes and because there is no information matrix to invert, specification searches are not as important to ANN.

In sum, research into hedonic pricing models should argue in favor of nonlinear modeling strategies—artificial neural networks are but one such, easily implemented, method that is relatively neutral to the many data problems that plague OLS. The bottom line in this paper is that pricing errors in linear models are significant, avoidable, and therefore costly and that data problems exacerbate these costs.

Appendix

Backpropagation[8]

Referring to Exhibit 1 and Equations (2) through (4), we wish to iterate on the network's weights (i) for each node (j) in layer (s):

$$w^s_{i,j}(k+1) = w^s_{i,j}(k) - \alpha \frac{\partial F}{\partial w^s_{i,j}}, \tag{A1}$$

using a gradient descent algorithm that minimizes a loss function given by:

$$F = {}_w \min \sum (t-a)^2, \tag{A2}$$

where t is a scalar target (i.e., value of the dependent variable) and a is the scalar network output (prediction). The simple feedforward network design pictured in Exhibit 1 has nine inputs, a single hidden layer consisting of three nodes, and an output layer consisting of a single node. Beginning weight values are initialized randomly.

Let $\mathbf{W}$ denote a 3×9 matrix of weights and $\mathbf{b}$ a 3×1 vector of biases. In the first layer, the nine inputs $\mathbf{p}$ are weighted and summed by each hidden node. These weighted sums (a 3×1 vector), $\mathbf{n}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}$, are then transformed using the arctangent function ($\mathbf{f}^1$), which compresses each sum into the interval $\{-1,1\}$, producing outputs $\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b})$. These three values represent the signal strength at each hidden node.

The values in $\mathbf{a}^1$ are then fed forward to the second layer. The second layer then weights and sums the first layer output, where $\mathbf{W}^2$ is a 1×3 vector of weights.

$$\mathbf{n}^2 = \mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b})) = \mathbf{W}^2(\mathbf{a}^1). \tag{A3}$$

This is a scalar that is also transformed and compared to the target value. Since $\mathbf{f}^2$ is linear in our network, a is the weighted sum of hidden layer outputs, i.e., the scalar:

$$a = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1) = \mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b})). \tag{A4}$$

It is this value that enters the loss function in (A2). The weight-adjustment process first feeds the input forward to the output layer and then propagates the errors back through the network using steepest descent.

To see this, consider again the gradient from (A2). With no loss of generality, we can ignore the bias $\mathbf{b}$ and the learning rate α in what follows. Assume there are m layers in the network so that $s = 1,...,m$. Using the chain rule and differentiating, the final layer gradient is:

$$\frac{\partial F}{\partial w_{i,j}^m} = \frac{\partial F}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m} = \frac{\partial F}{\partial n_i^m} \cdot a_j^{m-1}. \tag{A5}$$

The change in the gradient due to alteration in the weight for the j[th] input on node *i* in layer *m* is a function of the product of the *sensitivity* of the loss function to the j[th] input on that node, $\partial F/\partial n_i^m$, and the output from the previous layer at that node. Notice the interconnectivity of nodal output across layers. Using (A2), (A3), and recalling that, since the transfer function in the final layer is linear in our model, $a^m = n^m$, we can rewrite (A5) as:

$$\begin{aligned}
\frac{\partial F}{\partial w_{i,j}^m} &= \frac{\partial F}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m} \\
&= \partial \left[\frac{\sum_{j=1}^{sm} (t_j - a_j)^2}{n^m} \right] \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m} \\
&= -\left[2(t_j - a_j) \cdot \frac{\partial a_j}{\partial j_j^m} \right] \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m} \\
&= -[2(e_j)] a_j^{m-1}.
\end{aligned} \tag{A6}$$

The last term, a_j^{m-1}, is the result of differentiating (A3). In our network, this simplifies further since there is only a single node in the output layer ($e_j = e$). Therefore, we can drop the subscripting on *j* resulting in:

$$\begin{aligned}
\frac{\partial F}{\partial w_j^m} &= -[2(e)] a_j^{m-1} \\
&= s^m a_j^{m-1},
\end{aligned} \tag{A7}$$

where the term $s^m = \partial F/\partial n_i^m$ measures the sensitivity of the loss function. Continuing backwards through the network, at layer $m-1$, we have:

$$\begin{aligned}
\frac{\partial F}{\partial w_{i,j}^{m-1}} &= \frac{\partial F}{\partial n_i^{m-1}} \cdot \frac{\partial n_i^{m-1}}{\partial w_{i,j}^{m-1}} \\
&= \left[\frac{\partial F}{\partial n_i^m} \frac{\partial n_i^m}{\partial n_i^{m-1}} \right] \cdot \frac{\partial n_i^{m-1}}{\partial w_{i,j}^{m-1}}.
\end{aligned} \tag{A8}$$

From (A6), it is clear that $\partial F/\partial n_i^m = -2(e_j)$. Using the following definition:

$$\frac{\partial n_i^m}{\partial n_j^{m-1}} = \frac{\partial\left(\sum w_{i,j}^m a_j^{m-1} + b^m\right)}{\partial n_j^{m-1}} = w_{i,j}^m \frac{\partial a_j^{m-1}}{\partial n_j^{m-1}}, \tag{A9}$$

we write the value of the gradient as:

$$\begin{aligned}\frac{\partial F}{\partial w_{i,j}^{m-1}} &= -[2(e)]\left(w_{i,j}^m \frac{\partial a_j^{m-1}}{\partial n_j^{m-1}}\right) a_j^{m-2} \\ &= s^{m-1} a_j^{m-2}.\end{aligned} \tag{A10}$$

This is a recursive relationship that clearly propagates the error back through the network. Equation (A7) shows that the final layer weights change in relation to the size of the error, scaled by the output from the previous layer. From Equation (A10), the weights in the previous layer ($m-1$) carry these weight adjustments backwards through the network, scaling them yet again by the network's sensitivities at each node.

If we define $\partial F/\partial_i^m$ as s^m, we can rewrite the network learning algorithm:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s^m (a^{m-1})^T. \tag{A11}$$

The weight vectors are therefore adjusted iteratively along the gradient using steepest descent to locate a minimum squared error loss. A more rigorous derivation can be found in Haykin (1999).

Endnotes

[1] See, for example, Do and Grudnitski (1992), Worzala, Lenk, and Silva (1995), Goodman and Thibodeau (1998), and Nguyen and Cripps (2001) for which price is linear in the number of bedrooms and baths and where property age enters the regression up to the fourth power.

[2] Our data set spans 20 different townships and 19 different planning jurisdictions. Incorporating this information without aggregating townships, for example, would often lead to rank failure.

[3] Their training set consisted of 217 properties with a hold-out sample of 71 properties. Do and Grudnitski (1993) studied 105 properties while Tay and Ho (1992) trained on

833 properties and tested their network on a hold-out sample of 222 properties. Our data base consists of 46,467 properties spanning 1999–2005.

[4] We also tested the semi-log form in which the dependent variable was the natural log of sales price. This model uniformly underperformed the linear model. Semi-log forms are equivalent to $y = \exp(X\beta + \varepsilon)$, which is a restriction on the functional form—in this case, a particular nonlinear specification.

[5] We test a two-tailed alternative with degrees of freedom $pN - 1$ or $(1 - p)N - 1$.

[6] Volatility increased 35% from 1999 to 2005.

[7] MAPE is formally defined by $(\sum_{i=1}^{N}|P_i - \hat{P}_i|/P_i)/N$, where P is actual price and $\hat{P}$ is the predicted price.

[8] We borrow from the notation in Hagen, Demuth, and Beale (1997).

References

Allen, W.C. and J.K. Zumwalt. Neural Networks: A Word of Caution. Unpublished Working Paper, Colorado State University, 1994.

Bagnoli, C., B. Smith, and C. Halbert. The Theory of Fuzzy Logic and its Application to Real Estate Valuation. *Journal of Real Estate Research*, 1998, 16:2, 169–200.

Baen, J. S. and R.S. Guttery. The Coming Downsizing of Real Estate: The Implications of Technology. *Journal of Real Estate Portfolio Management*, 1997, 3:1, 1–18.

Detweiler J.H. and R.E. Radigan. Computer-Assisted Real Estate Appraisal: A Tool for the Practicing Appraiser. *The Appraisal Journal*, 1996, 91–101.

Do, A.Q. and G. Grudnitski. A Neural Network Approach to Residential Property Appraisal. *The Real Estate Appraiser*, 1992, 58, 38–45.

——. A Neural Network Analysis of the Effect of Age on Housing Values. *Journal of Real Estate Research*, 1993, 8:2, 253–64.

Garson, G.D. Interpreting Neural-Network Connection Weights. *Artificial Intelligence Expert*. 1991, 6, 47–51.

Goodman, A.C. and T.G. Thibodeau. Age-Related Heteroskedasticity in Hedonic House Price Equations. *Journal of Housing Research*, 1995, 6, 25–42.

Grether, D. and P. Mieszkowski. Determinants of Real Values. *Journal of Urban Economics*, 1974, 1:2, 127–45.

Guan, J., J. Zurada, and A.S. Levitan. An Adaptive Neuro-Fuzzy Inference System Based Approach to Real Estate Property Assessment. *Journal of Real Estate Research*, 2008, 30: 4, 395–422.

Hagan, M.T., H.B. Demuth, and M. Beale. *Neural Network Design*. First edition. Massachusetts: PWS Publishing Co., 1997.

Haykin, S. *Neural Networks: A Comprehensive Foundation*. Second edition. New Jersey: Prentice Hall, 2003.

Huang, C-S., R.E. Dorsey, and M.A. Boose. Life Insurer Financial Distress Prediction: Neural Network Model. *Journal of Insurance Regulation*, 1994, 13:2, 131–67.

Intrator, O. and N. Intrator. Interpreting Neural-Network Results: A Simulation Study. *Computational Statistics and Data Analysis*, 2001, 37:3, 373–93.

Kahneman, D. and A. Tversky. Subjective Probability: A Judgment of Representativeness. *Cognitive Psychology*, 1972, 3, 430–54.

Nguyen, N. and A. Cripps. Predicting Housing Value: A Comparison of Multiple Regression Analysis and Artificial Neural Networks. *Journal of Real Estate Research*, 2001, 22:3, 313–36.

Rossini, P.A., P.J. Kershaw, and R.R. Kooymans. Microcomputer Based Real Estate Decision Making and Information Management—An Integrated Approach. Paper presented at the Second Australasian Real Estate Educators Conference, 1992.

Rossini, P.A., P.J. Kershaw, and R.R. Kooymans. Direct Real Estate Analysis—The UPmarket™ Approach to Real Estate Decision Making. Paper presented at the Third Australasian Real Estate Educators Conference, 1993.

Shiller, R.J. and A.N. Weiss. Evaluating Real Estate Valuation Systems. *Journal of Real Estate Finance and Economics*, 1999, 18:2, 147–61.

Thaler, R.H. Mental Accounting and Consumer Choice. *Marketing Science*, 1985, 4, 199–214.

Tsukuda, J. and S.I. Baba. Predicting Japanese Corporate Bankruptcy in Terms of Financial Data Using Neural Networks. *Computers & Industrial Engineering*, 1994, 27:1–4, 445–48.

Tay, D.P.H. and D.K.H. Ho. Artificial Intelligence and the Mass Appraisal of Residential Apartments. *Journal of Property Valuation and Investment*, 1992, 10:2, 525–39.

Worzala, E., M. Lenk, and A. Silva. An Exploration of Neural Networks and Its Application to Real Estate Valuation. *Journal of Real Estate Research*, 1995, 10:2, 185–201.

Steven Peterson, Virginia Commonwealth University and Virginia Retirement System, Richmond, VA 23218-2500 or speterson@varetire.org.

Albert B. Flanagan, Williams Appraisers, Inc., Raleigh, NC 27607 or benji@williamsappraisers.com.