Automated Modeling and Nonlinear Axis Scaling

Leejay Wu

CMU-CS-05-145

May 2005

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christos Faloutsos, Chair
Anastassia Ailamaki
Andrew Moore
Richard Caruana, Cornell University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under Grant No. IIS-9910606. Additional material support was provided by Intel.

Keywords: scaling, modeling, feature selection

Dedicated to my family.

Abstract

This thesis examines nonlinear axis scaling and its impact on the modeling of inter-attribute relationships. Through automated methods, the described system identifies possible scaling methods; decides which attributes serve as inputs or outputs; and builds regression trees that quantify these relationships. While the experiments focus on the accuracy and complexity of these models, both of which one can attempt to quantitatively examine, the results also consider applicability towards the inherently more qualitative task of rule-based outlier or anomaly detection. The results demonstrate that the use of nonlinear axis scaling, even in an automated system, can provide significantly more accurate models compared to the unscaled case without proportionally higher complexity costs; and also can help reveal unusual tuples in which what is unusual is not any individual value, but the combination thereof.

Acknowledgments

The author also owes a debt to his longtime advisor, Christos Faloutsos; and others including Anastassia Ailamaki, Anthony Brockwell, Hal Burch, Richard Caruana, Deepayan Chakrabarti, Elena Eneva, Jun Gao, George Matcuk, Andrew Moore, Robert Murphy, Raymond Ng, Menghzi Wang, and the floating membership of the CMU Database Group, for feedback, counsel, data, and otherwise helping me get through it all.

The patience of the department as a whole should also be noted – faculty, staff, other students.

Most of all, the author appreciates the patience and support of his family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis focuses on automatically finding mathematical relationships between the attributes of mostly-numerical data sets and expressing them through accurate and compact functions. Furthermore, the described system is designed to search for a mutually independent set of attributes for model inputs, rather than to require the user to specify which attributes should be modeled with which others. In addition, it does not assume that the original axis scales are appropriate for the model task, but uses uniformity heuristics to identify suitable nonlinear axis scalings for both independent input and dependent output attributes. Finally, the most accurate models are then used to assist the search for interesting exceptions, or model-based outliers.

Such functions may be directly used in prediction or classification. One obvious example comes from the insurance business, since insurance companies need to estimate their risk incurred when covering potential customers based on their histories and current behavior. An auto insurance company might consider such factors as sex, age, previous history in terms of accidents or moving violations, occupation, the city of residence, or the model of automobile in deciding how much coverage they can extend and what premium to demand. Passenger airlines and their resellers are well-known for charging fares that differ significantly even within the same flight and cabin class; the prices they charge likely depend not only on expenses such as the costs of aviation fuel, aircraft maintenance and employing air crew but also the prices charged by competitors, the availability of seating, the days remaining until the flight, and the method used to purchase the ticket. In a market in which coach-class tickets on competitive routes are essentially commodities, it would help to know the relationship between fares and demand. The more fully understood such a relationship is, the easier it would be to define the related optimization problem.

Less obviously, there exist benefits besides predictions from building models that describe the relationships between attributes. For instance, if a model accurately predicts data exploiting a non-obvious relationship, it might be worthwhile to examine the possible causes. In cases where collecting data is expensive, potentially harmful, or has other drawbacks, examining a model may also allow one to narrow the scope of data collection if it performs well without using all attributes.

In addition, when strong relationships hold for most of the data, the few exceptions may be of keen interest. One example comes from the `abalone` data set hosted by the UCI Repository [Murphy and Aha, 1994]. The data cover the physical attributes of individuals from a particular variety of shellfish. The documentation suggests that the number of rings – which varied from one to twenty-three – in the shell of an abalone, added to 1.5, approximates its age. There's also a gender attribute, specified as male, female or infant. One might expect infant abalones to be young; however, the models reveal a number of strange tuples including one allegedly corresponding to a 20.5-year-old infant abalone. Neither its age nor its gender is by itself strange, judging more from the data set rather than any knowledge regarding shellfish, but their combination would seem to suggest either a data collection or entry error, or developmental abnormalities. This is the type of anomaly which we would expect an otherwise accurate model to detect through unusually high prediction error.

ith data where most tuples consist of innocent behavior while a few may be adversarial, such as a database of mostly honest borrowers while a few plan to defraud the lender, rules which hold for normal behavior but not for the adversarial instances would be extremely useful. In other domains, the opposite situation may hold in which most data do not fit well with certain rules unless they're exhibiting interesting properties; for instance, if a pattern exists for system failures, adherence to that pattern may be much more interesting than the opposite. In such a situation it may be useful to search for rules on the anomalous cases, and exclude those which also seem to hold for too many of the less interesting cases. This latter procedure is difficult without labeled examples or domain expertise, however.

This work considers data mining from the perspective of model building in an unsupervised domain, where the term 'model' specifically refers to any equation that approximates individual values of individual attributes as functions of the associated values of other attributes; and 'unsupervised' means that the system operates without requiring input-output labels or being able to request additional data. The system is also not permitted to assume that the original scales of the attributes are optimal for its purposes; for instance, it may be the case that an accurate model of acceptable complexity would be easier to build if a logarithmic transformation were applied to an attribute. To deal with this possibility, we must consider a space of possible nonlinear axis scalings and attempt to determine which will help the model-building process.

Consider the following example: the area and population of regions as defined by the 2001 edition of the CIA World Factbook. Prior to any nonlinear scaling, the set looks like Figure 1.1; after nonlinear but order-preserving and invertible scaling on both axes as selected in experiments described in a later chapter, we get Figure 1.2. The latter seems more conducive to demonstrating both the existence of a relationship between area and population, and in addition noting the presence of regions that have unusually low or high population densities.

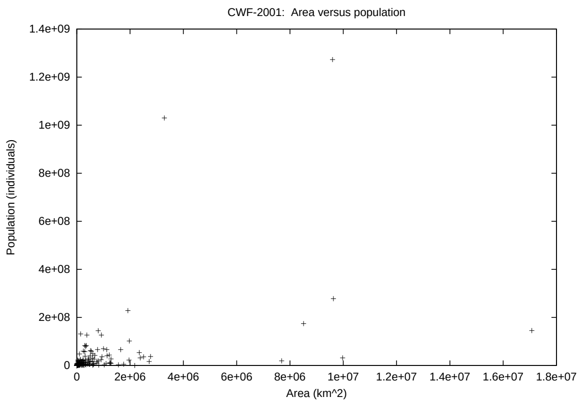


Figure 1.1: Area and population data from the 2001 edition of the CIA World Factbook.

The excerpts shown in Figures 1.3 and 1.4 show that the incomprehensibility of Figure 1.1 is not merely due to a few outliers causing most of the data to fall within a small region suitable for examination with a microscope; focusing only on the densely populated region one arrives at Figure 1.3, which resembles the original graph. Repeat the discarding and you reach 1.4 which again has a dense region near the origin and not much more clarity than before. Clearly, this is not merely a problem of a neat data cluster obscured by a few outliers; likewise, neither affine transformations nor axis rotations would suffice. We therefore consider nonlinear axis scaling – which frequently can help, and in this case does – a crucial instrument.

The current system may be divided into three major components: nonlinear

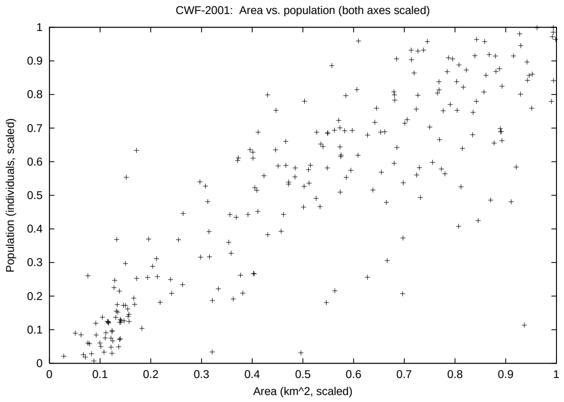


Figure 1.2: Area and population data from the 2001 edition of the CIA World Factbook, after nonlinear axis scaling.

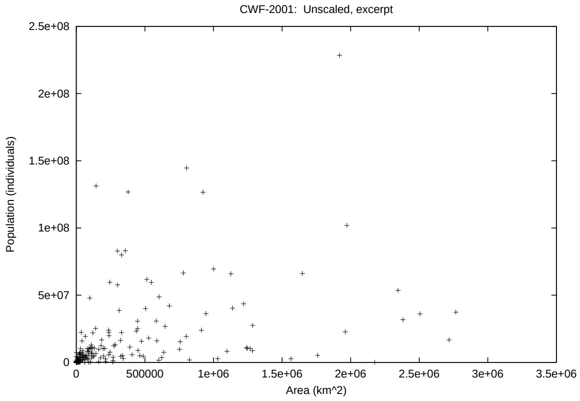


Figure 1.3: After discarding "outliers" by cropping sparsely-populated extremes in the unscaled 2001 CIA World Factbook data.

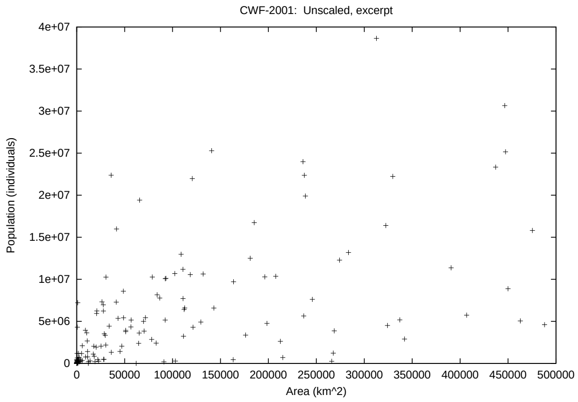


Figure 1.4: After another round of discarding “outliers” and cropping.

scaling, input attribute selection, and modeling. Each stage produces output that may be of interest by itself; for instance, the choices of scalings or the division between inputs and outputs may be of separate interest. However, the modeling stage is the most amenable to quantitative analysis as we can assess the accuracy of the models more easily than we can the benefits of knowing that a particular attribute seems best scaled in a certain fashion, and is the most direct means of justifying the previous stages; therefore, the experiments and analysis thereof focus on this aspect rather than making more direct assessments of the selected axis scalings or choice of inputs and outputs.

Chapter 2

Design Questions

As noted in the previous chapter, this work describes a system designed at a data mining in a unsupervised environment. We have tables of mostly numerical data, where rows and columns correspond to instances and attributes; but we do not necessarily have input-output labels. In addition, it may be easier to build a more accurate model after nonlinearly scaling one or more if the attributes.

Ideally, such axis scalings, the determination of inputs and outputs, and the resulting models should be built *automatically* with a minimum of tuning, so as to supplement the useful but less-scalable expertise and attention of any human domain expert. We want a fully-automated system for identifying the relationships between inputs and outputs, and for evaluating how well data fits these relationships so we can find interesting exceptions, when given a complete data set consisting primarily of numerical attributes.

2.1 Four fundamental questions

In light of the above, for any given set we might consider the following questions.

1. What scaling is appropriate for each axis?
2. Which attributes should we model?
3. Which attributes should be the inputs for those models?
4. How should these models be chosen?

Of these, first examine the second and third questions. These closely-related queries stem from the fact that to fit a function requires both inputs and outputs. For simplicity, let us restrict the solution space by requiring that each attribute be either an input or an output, but not both. In addition, let us suppose that our model builder can determine which attributes are most relevant among possible inputs. This reduces the second and third questions to the problem of dividing the set of attributes between inputs and outputs.

The fourth question also directly relates to the modeling problem. Even if one has been given a fixed partition between inputs and outputs, one still needs to decide how to model. The exact choice of the model class and the method for searching that specific class matters. For instance, consider the problem of modeling a data set that follows the relationship $y = \sin(x)$. For this data, a model class featuring linear combinations of sine waves parameterized by amplitude and phase shift could be expected to yield a solution with more accuracy and fewer parameters than a model class composed of square waves, which might in turn do better in terms of accuracy than a class of non-periodic functions such as piecewise linear models. While for a finite data set all three of these classes could be used to approximate the data with excellent accuracy, again efficiency might vary.

As noted above, one needs not only a model class but also a method for searching that class. This complicates the design of fully automated systems. For instance, while the weights in a neural network can be trained through methods such as back-propagation, choosing the initial design in terms of nodes, activation functions and connectivity is considerably more difficult. It may not even be obvious how many times one should re-initialize and run until convergence.

Lastly, consider the first of the four questions – what to do about scaling. In theory, if all one cared about were attaining precise and accurate models, many model classes possess sufficient flexibility that scaling would not matter. In practice finding optimal or even good models within sufficiently flexible model classes happens to be quite difficult; thus, scaling can affect the end result even if the model class itself contains accurate models for the unscaled data. In addition, scaling might affect the efficiency of the models. Consider a two-attribute finite data set following the formula $y = e^x$. Even a piecewise linear model would work – but depending on the range of x values, it might require a substantial number of line segments, and therefore a similarly high number of parameters. In addition, such a model would not generalize to x, y tuples well outside the original range despite the simplicity of the underlying rule. On the other hand, the application of a logarithmic transformation $\hat{y} = \log y$ results in $x = \hat{y}$ – a simpler relationship. This simplifying of the relationship, however, does come at the cost of also scaling the noise distribution. It is possible that in the unscaled case the noise obeyed something like a simple Gaussian distribution, which after scaling would not even be symmetric, let alone Gaussian. Scaling might also have an impact on the efficacy of input selection depending on the methods used.

Chapter 3

Related work

Between attempts to handle the curse of dimensionality and the desire to find patterns and models, a substantial body of work proves relevant to the problems at hand.

3.1 Dimensionality reduction

The problem of dividing a set of attributes into two disjoint sets, one of model inputs and one of model outputs, seems related to the well-explored problem of dimensionality reduction. Methods which label some attributes as worth keeping and others not, provide such a binary partition usually based on some notion of independence or utility.

This thesis does not explore methods in which the same attribute may be used for inputs and outputs; however, clustering in local subspaces and subsequent partitioning of the data set on row-based boundaries would provide at least some of this functionality. Methods which perform dimensionality reduction on subsets such as [Chakrabarti and Mehrotra, 2000], which performs clustering; [Agrawal et al., 1998] and [hung Cheng et al., 1999], which look for clusters in subspaces; and [Keogh et al., 2001], which uses contiguous regions within the temporal domain, all seem relevant.

Among more traditional dimensionality reduction methods, those based upon linear projection such as principal components analysis [Bartell et al., 1992] [Calvo et al., 1998] [Jolliffe, 1986] [Partridge and Calvo, 1998], random linear projection [Dasgupta and Gupta, 1999], and independent components analysis [Hyvärinen et al., 2001]

[Hyvärinen, 1999] generate linear combinations of inputs rather than a partitioning. The same holds for many nonlinear projection methods, such as principal curves [Chang and Ghosh, 1998], kernel PCA [Schölkopf et al., 1998], neural network methods [Baldi and Hornik, 1989] [DeMers and Cottrell, 1993] [Jones, 1992] [Kramer, 1991] [Takahashi and Tokunaga, 1999], locally linear embedding [Roweis and Saul, 2000], and mapping methods like FastMap [Faloutsos and Lin, 1995], Kruskal's multidimensional scaling [Kruskal, 1964], Sammon maps [Sammon, 1969], Isomap [Tenenbaum et al., 2000] and MetricMap [Wang et al., 1999].

We therefore restrict our search to feature selection methods. The maximum entropy discrimination approach [Jebara and Jaakkola, 2000], provides a method designed to act in accordance with some pre-existing goal such as classification or rejection in the supervised case; that is, labels are both useful and necessary. In this context, people have examined the merits of directly using labels to influence the selection process [Kohavi and John, 1997] [Tsamardinos and Aliferis, 2003]. Another method, generalized LASSO [Roth, 2002], selects relevant attributes in the domain of gene microarrays. The generalized LASSO algorithm differs from our problem domain in that it focuses on probabilistic classification rather than continuous prediction.

In the domain of text information retrieval, some work has been performed identifying insignificant terms without requiring labels [Lang, 1995]. There, however, vectors consist solely of frequency counts, which means that the data lie entirely within the set of non-negative integers with rather similar meanings. In other domains, different attributes may have wildly different scales and semantics.

The method of [Traina Jr. et al., 2000] seems appropriate as a starting point as it deals with individual attributes and attempts to capitalize on even nonlinear dependencies between them, without requiring labels. This algorithm and alterations thereof shall be described in more detail later as it comprises an integral part of the overall system.

3.2 Models

In addition to the neural network methods mentioned above, which perform function-fitting in addition to possible nonlinear dimensionality reduction; and the principal components methods (linear or otherwise) which also relate features to each other through the necessary mathematical derivations; one can apply other algorithms such as genetic programming methods [Choenni, 2000], and assorted regression tree methods such as CART [Breiman et al., 1984], SECRET [Dobra and Gehrke, 2002], C4.5 [Quinlan, 1993] and RT [Torgo, 1999]. Rule-based regression function prediction methods have also been explored, such as in [Weiss and Indurkhya, 1995]. Such methods may be suitable after using feature selection or remapping techniques that do not provide mathematical grounds for representing old features in the new space.

3.3 Model complexity

In the most general theoretical, optimistic, superficial sense, function fitting consists of accepting inputs and accurately computing the corresponding outputs. However, this absolutist formulation seems problematic for a variety of reasons. Among the more obvious objections, there exists the possibility that noise, measurement or data entry errors, nondeterminism, and factors not captured in the inputs will prevent a perfect correspondence between inputs and outputs. Furthermore, even if one instead aims for "the most accurate model possible, after data cleansing", with accuracy measured by some error metric, absurdities still present themselves as solutions as per Figure 3.1.

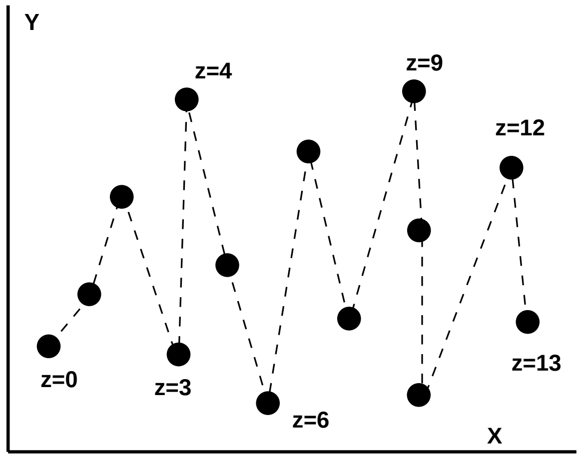


Figure 3.1: If you ignore complexity, you risk approving dimensionality reduction or modeling through connecting the dots, where a single parameter plus a very complex curve are used to approximate as high a dimensionality data set as you'd like.

There is a significant body of philosophically related work, all of which incorporate some form of Ockham's Razor in the sense that they quantify the trade-off between accuracy and complexity. From the algorithmic complexity and compression fields, we find theories regarding Kolmogorov complexity, Solomonoff complexity, minimum message length, minimum description length, and similar methods [Hansen and Yu, 2001] [Rissanen, 1978] [Wallace and Dowe, 1999]. Simpler methods such as the Bayesian and Akaike information criteria – BIC and AIC respectively – have also been proposed [Akaike, 1974]. All of these methods have been applied towards selecting models while taking into account model complexity. For instance, the simplest approaches, AIC and BIC, suggest selection from among the members of a model class in order to minimize

$$AIC(\theta) = \frac{n}{2} \log \epsilon + |\theta|$$

or

$$BIC(\theta) = \frac{n}{2} \log \epsilon + \frac{|\theta|}{2} \log n$$

with model parameter vector θ and number of instances n.

The more complex methods may be more applicable when considering a variety of model classes with different properties. For instance, one algorithmic complexity approach frames the problem as the determination of a two-part string of minimum size that, provided to a specific universal Turing machine (UTM), returns the exact symbol string we wish to model or compress. Whether or not the UTM needs to halt immediately after properly outputting the string varies per from definition to definition.

In the two-part framework, the input string needs to be composed of two consecutive non-intersecting parts with different purposes. The first part encodes a hypothesis for the UTM, while the second part defines parameters required to generate this input from that hypothesis. More specific conditions formalizing this partitioning scheme may be found in [Wallace and Dowe, 1999]. One may also use one-part schemes which do not explicitly differentiate between the encoding of the hypothesis and specific inputs. In either case, the objective remains minimizing the length of the complete input string.

The brief discussion above neglects certain practical issues, such as whether or not such methods are computable – in the general case determining the exact Kolmogorov complexity is not – and how one designs the initial UTM. With regards to the second issue, we note that the choice of UTM affects the solution. In fact, the problem of choosing the UTM resembles the dilemma of choosing model classes; in both cases, just because different choices may each be capable of modeling the data does not mean that they actually will do so with similar efficiency, accuracy and ease.

A recent approach by Keogh, Lonardi and Ratanamahatana [Keogh et al., 2004] replaces the UTM selection problem involved in Kolmogorov complexity with the problem of selecting lossless compression methods such as the algorithms behind `gzip` or `bzip2`, and uses these methods to approximate algorithmic complexity. For continuous, real-valued numbers the authors suggest discretization.

As with dimensionality reduction, significant differences appear between the problem we have in mind and the problem covered by the aforementioned methods of comparing models. Specifically, we wish to model unordered sets of vectors that may contain either integers or floating-point numbers, instead of a single sequence of symbols. In addition, we expect errors and imperfect models on realistic data; the problem is modeling, not lossless compression. Furthermore, we focus on the values of the data, and not their representations; we find relationships between attributes of greater interest than the relative frequency of bit strings used in a particular serialization of the data as encoded in a specific format in memory.

3.4 SPARTAN

The SPARTAN system comprises the previous work closest to our problem [Babu et al., 2001]. SPARTAN first builds Bayesian networks to identify dependencies. A Bayesian network is simply a directed acyclic graph with exactly one node per attribute, and where each node contains a probability distribution for the corresponding attribute conditioned on the attributes whose nodes have edges leading to it. A Bayesian network, then, expresses dependencies without requiring an explicit instantiation of the full joint probability table. The algorithm used by SPARTAN returns usable ones within $O(m^4)$ given m attributes [Cheng et al., 1997]. Taken as a whole, SPARTAN in practice scales approximately linearly with the number of tuples sampled [Babu et al., 2001].

Bayesian nets assist the search for models for each attribute by suggesting dependencies. For any attribute A and it corresponding vertex, define the Markov blanket as the set containing its parents, its children, and the other parents of those children. Figure 3.2 show a simple example. Given full information on the attributes in the Markov blanket, attribute A will be independent of all the other attributes. Thus, when considering how to model A, a reasonable start is to first look at the Markov blanket.

Instead of being used directly for modeling, the Bayesian network leads to another graph. In this graph, the vertices again represent attributes. Undirected edges connect vertices wherever SPARTAN would consider one a candidate predictor of the

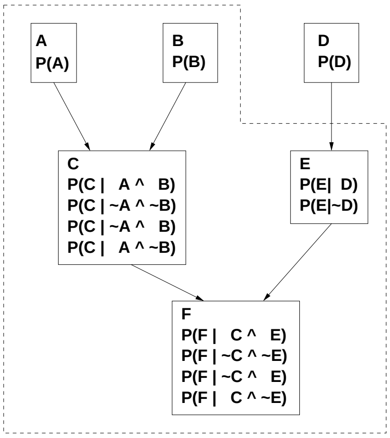


Figure 3.2: A Bayes net for Boolean attributes A, B, C, D, E, F. A, B and D are independent of all other attributes, while probability tables exist for C in conditioned on A and B; E, conditioned on D; and F, conditioned on C and E. The Markov blanket for C consists of all attributes except D; A and B are parents of C, F is a child of C, and E is a parent of F.

other – for instance, when one attribute references the other as part of the former's Markov blanket. Thus, the Bayesian network serves to limit the connectivity of this second graph.

This limitation proves important because initially, the final phase treats the second graph as a weighted maximum independent set (WMIS) problem in which the weight of each vertex reflects the immediate gain from modeling the corresponding attribute using its neighbors. However, unlike ordinary WMIS, SPARTAN edits the graph while it iterates to permit models to be generated using inputs that originally lacked adjacency to the attributes to be modeled. For instance, an attribute A need not be materialized just because SPARTAN chose a neighbor B to be predicted, so long as the remaining materialized neighbors of A plus the still-materialized neighbors of B can themselves provide a sufficiently accurate and efficient model.

For the underlying model, SPARTAN uses piecewise-constant CART trees [Breiman et al., 1984]. These classification and regression trees provide a divide-and-conquer approach which allows significant flexibility in modeling, at the cost of increased complexity. Here, the regression trees use single-attribute tests in the internal nodes and constant labels in the leaves. To maintain strict error tolerances, SPARTAN also stores outliers in the leaves, where SPARTAN defines outliers as data instances for which the proposed model, given the corresponding inputs, would produce values whose error exceeded tolerances. The user specifies these tolerances as fractions of each attribute's range.

SPARTAN also makes use of "fascicles" [Jagadish et al., 1999] in order to further compress the materialized, or predictor, attributes. Again, SPARTAN takes care to ensure that this lossy compression does not cause cascading errors to exceed error tolerances.

For assessment purposes, SPARTAN uses storage cost – feasible as it assumes discrete data, thus enabling the use of prefix-free codes. The cost model ignores errors within tolerances, while errors beyond them accrue cost due to the need to explicitly store outliers. The CART trees themselves require storage. For every node, one must store data that identifies the type of the node as well as any test data such as which attribute gets tested against what value, and the labels of the child nodes corresponding to positive and negative results of the test – or predicted values. Since SPARTAN only uses one class of models, CART trees, and the system focuses on semantic compression, more complicated assessment schemes may be inappropriate.

3.5 Fasicles and ItCompress

Fascicles [Jagadish et al., 1999] and the more recent ItCompress [Jagadish et al., 2004] provide alternative semantic compression methods, where they define "semantic" compression as compression that focuses on the values within the numerical data rather than the bit strings that represent those values for any particular encoding scheme.

Both systems produce a compressed representation in which references to a dictionary of tuples stand in for individual rows. With fascicles, tuples whose compressed representations do not meet error tolerances need to be explicitly stored as outliers. ItCompress refines this by storing a bitmap for each tuple that specifies which attributes' values did not meet error tolerances so that individual values instead of entire tuples get stored as outliers. Both methods focus on compressing finite, static sets and do not attempt to derive models useful for prediction. This focus limits the amount of information that one can learn from running either system; for instance, the dictionary of representative tuples does not provide rules governing relationships between the attributes. Thus, while from a pure compression perspective ItCompress may perform well even compared to model-based methods like SPARTAN [Jagadish et al., 2004], from a data mining perspective the latter may be a more useful and informative choice.

Chapter 4

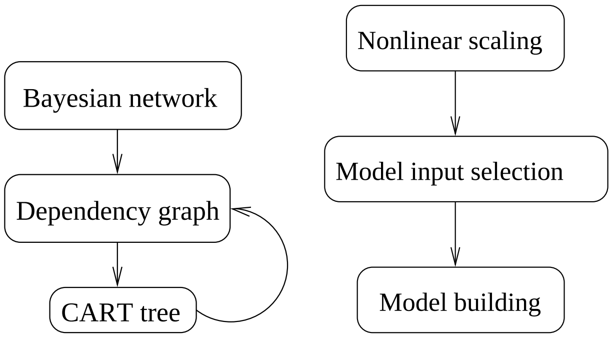
System Design

Consider the system as designed and implemented. The system consists of three stages – scaling, selection and modeling. Each of these stages processes the data with the intent of eventually returning efficient, accurate models demonstrating the relationships between attributes, and may return additional information regarding the nature of the data. The name *Briareus* seems somewhat appropriate; the system tries numerous possible axis scalings for each numerical attribute, and does not commit to any particular axis scalings for model outputs until their relative impact can be assessed during model selection.

However, unlike SPARTAN (Figure 4.1(a)), this system (Figure 4.1(b)) separates input-output determination from modeling. SPARTAN involves more complexity in determining inputs and outputs, and allows the generated models to affect its labels; this thesis project uses a simpler workflow of splitting the attribute set once into inputs and outputs without first generating models, but introduces complexity through the use of nonlinear scaling as well as a more complicated model class. Section 4.4 gives a more detailed qualitative comparison of the two systems.

4.1 Axis scaling

How should one scale data? In practice, this question has sometimes been entirely ignored; for instance, the SPARTAN system operates on the data as-is [Babu et al., 2001], which is reasonable when using piecewise-constant models such as CART trees. It may also be sometimes be the case that we have *a priori* knowledge that the current axes suffice for model-building. However, in the general case we cannot



(a) SPARTAN (b) Briareus

Figure 4.1: High-level diagrams of SPARTAN and the described system, Briareus.

assume such. For instance, recall the area and population data from the 2001 edition of the CIA World Factbook. In the unscaled data of Figure 1.1 is unclear whether there is any relationship between area and population; the scaled version in Figure 1.2 suggests both the existence of some such relationship, and also the presence of a non-trivial number of cases which deviate from it. As described earlier, this is an example of a data set for which nonlinear axis scaling will be of far more help than affine transformations, axis rotations, or clustering and outlier detection in the unscaled space.

Adding nonlinear scaling to the modeling problem obviously complicates the search space. While the above example shows that nonlinear scaling can help, to actually perform it one must not only declare a set of nonlinear scalings to consider, but also to describe a process for selecting which methods to apply on specific attributes and specific subsets. The set of possible nonlinear scalings for even one attribute is infinite; similarly innumerable might be the possible criteria for preferring one subset over another. The particular choices made have been based on a pragmatic approach; rather than opt for an elegant minimalist approach, the axis scaling system aims for comprehensiveness through the following computationally intensive but highly flexible approach.

Briareus currently limits the realm of possible scalings to the cross-product of two finite sets – one set of Box-Cox transformations, and one set of cumulative distribution functions selected by estimation on the data – plus the identity function. For instance, one possible scaling function that can be returned is the composition of the logarithmic function, a Box-Cox transformation, and the cumulative distribution function $F_{\mu,\sigma}$ of a given normal distribution:

$x' = F_{\mu,\sigma}(\log x)$

The rest of this section discusses these methods, their justification, and the methods for choosing among them.

Box-Cox transformations represent "traditional" nonlinear axis scaling methods. Since Box-Cox transformations require strictly positive values, a constraint not necessarily obeyed by the original data, the Briareus uses a two-parameter (λ, Δ) form as follows. For each numerical value x:

If $\lambda \notin \{0, 1\}$,

$$f(x) = \frac{(x + \Delta)^{\lambda} - 1}{\lambda}$$

If $\lambda = 0$,

$$f(x) = \log(x + \Delta)$$

If $\lambda = 1$,

$$f(x) = x$$

To permit iterative search, Briareus limits the set of λ values to a fixed set – arbitrarily defaulting to $\{0, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$ – and chooses Δ based on the minimum x_{min}. If $x_{\text{min}} < 10^{-10}$, Briareus sets $\Delta = 10^{-10} - x_{\text{min}}$; otherwise, $\Delta = 0$. These transformations provide a range between the identity function and the frequently available and chosen logarithmic transformation.

4.1.1 Reasons for using cumulative probability distribution functions

The cumulative distribution functions of univariate probability distributions, while rather less traditional choices for axis scaling, have the following desirable properties.

Monotonicity Just like Box-Cox transformations, univariate cumulative distribution functions (CDFs) possess monotonicity. For any given CDF $F(x)$, either $\forall a \forall b : a < b \rightarrow F(a) \leq F(b)$ or $\forall a \forall b : a < b \rightarrow F(a) \geq F(b)$, and in either case strict equality should be restricted to cases of zero density over $[a, b]$. This last condition can happen with functions distributions restricted to integer outcomes or to a given truncated range, or due to implementation issues such as the lack of arbitrary-precision arithmetic or the use of approximations. An monotonic axis scaling locally stretches or compresses an attribute, but preserves any ordering.

Likelihood of occurrence Popular probability distributions are usually thus because they model common behavior. For instance, the Gaussian or 'normal' distribution actually models real processes, such as the summation of a large number of independent identically distributed random variables. Scaling methods that take into account some possible generative process or family of such would be useful in identifying any instances; in addition, should such a distribution be correctly identified, the resulting cumulative distribution function should transform the data into something considerably more uniform. It seems possible that this would assist modeling enough to make CDFs a useful tool. Similarly, the Box-Cox transformations may be helpful for the same reasons that people study power laws.

Known estimation methods For many distributions there exist closed-form estimators. Some distributions require numerical methods; for instance, the well-known mixture of normals may be estimated using the iterative expectation-maximization method [Dempster et al., 1977], but no reasonable closed-form estimator exists. As long as these methods can be automated, they can be used. Some distributions have incremental estimators, which would be advantageous when considering stream data or even static sets when ordinary multi-pass methods would require too much memory. For instance, the Scalable EM algorithm incrementally estimates a mixture of univariate or multivariate normals. It may be also more practical than the original EM method on large databases, as it summarizes data in order to avoid repeated iteration over the original set [Bradley et al., 1999].

Known evaluation methods Among other methods, those seeking to evaluate how well a given distribution fits data can consider χ^2 testing, log-likelihood, Anderson-Darling, or the Bayesian information criterion (BIC). Again, Briareus needs to use automated methods instead of constantly prompting users for judgment calls. As implemented, the current system considers two metrics:

1. The A^2 value involved in computing Anderson-Darling scores,

$$A^2 = -N - \frac{\sum_{i=1}^{N}(2i-1)\left(\log F\left(x_i\right) + \log\left(1 - F\left(x_{N-i+1}\right)\right)\right)}{N}$$

with the data x_1 to x_N sorted in ascending order, and estimated cumulative distribution function F.

2. The correlation between the actual data and the predicted quantiles based on F^{-1} at 197 evenly distributed percentiles, 1% to 99% inclusive. Keeping the range thus avoids issues related to numerical approximations and certain distributions' percentage-point functions; not all such functions are defined at 0% and 100%, and the approximations may break down in the vicinity of those percentages.

Since the distribution parameters have been estimated from the data rather than as an *a priori* hypothesis, the A^2 value does not lead to a valid significance level from the usual tables of critical values. However, the raw values still provide a basis for comparison without requiring discretization, unlike methods aimed at discrete distributions such as the χ^2 test. Furthermore, since it uses the cumulative distribution function instead of the density, a single outlier seems unlikely to dramatically alter the results. This contrasts with likelihood-based tests such as the BIC, which risk being dominated by the low likelihood of a single extremely improbable value.

The scaling phase iterates through pairs of permitted Box-Cox transformations and probability distributions. Thus, it proves necessary to have an automated method for filtering out unacceptable scalings. For this, it uses both the A^2 value and the quantile-quantile correlation; the higher (worse) the A^2 value, the higher (better) the minimum acceptable correlation.

Estimates as summaries The choice of distribution and associated parameters can serve as a summary that informs the user of specific, perhaps useful properties. The partitioning of the summary into a scaling component and a distribution component means that each should be readily interpretable compared to their combination.

Flexibility Given enough components, a mixture of normals can approximate any continuous distribution within arbitrary positive error tolerances, while still having well-known and reasonably tractable estimation techniques such expectation-maximization [Bradley et al., 1999] [Johnson and Kotz, 1970] [Xu and Jordan, 1996].

4.1.2 Choosing among probability distributions

Ideally, the choice of distributions would reflect accurate priors in order to avoid unnecessary computation and produce more likely axis scalings. However, for the purposes of evaluation Briareus relies on a fixed list of distributions with an implied equal weighting; it does not currently take into account the relative likelihoods of different distribution classes beyond the binary weighting of whether or not a distribution is implemented and enabled. This seems appropriate when assuming the absence of domain knowledge. Instead, distributions were chosen with an eye towards flexibility, frequency, and ease of estimation.

Briareus uses only univariate distributions in order to support axis scaling. Unless otherwise noted, the parameterizations and estimators used match those found in [Johnson and Kotz, 1970]. Estimators exist for other distributions such as multivariate normals and mixtures thereof, but these do not nearly as readily lend themselves to attribute scaling or selection.

1. Exponential: While the exponential distribution could be subsumed by the also-included gamma distribution, the two-parameter exponential distribution used happens to be much easier to estimate.

2. Gamma: The system includes the gamma distribution by default because of its considerable flexibility. The implementation of this distribution employs the chi-square distribution, which in turn uses the normal distribution.

3. Generalized lambda: This implementation can only be considered partial, due to the incompleteness of the tables necessary for estimation [Karian et al., 1996]. As with the `gamma`, it was implemented for flexibility more than expected frequency.

4. Normal and variations thereof: Normals, doubly-truncated normals, mixtures of normals and mixtures of truncated normals are included. Ordinary normals can be estimated simply using the method of moments. Normals with up to two unknown truncation points, possibly both on one side of the mean of the underlying non-truncated normal, can be estimated using integration and simultaneous solving of nonlinear equations [Cohen Jr., 1950]; however, more robust estimation performance was observed when using a Levenberg-Marquardt search in order to minimize sum of squared deviations of estimated versus observed quantiles. Therefore, the described system uses the latter

method. Details and a brief empirical comparison may be found in Appendix A.

The mixture model families implemented arbitrarily limit the number of components to twenty to ensure termination within a reasonable time. In practice, this limit was never reached.

For mixtures of ordinary normals, Briareus uses the standard expectation-maximization algorithm [Dempster et al., 1977]; if scalability or incremental updates were more of an issue methods such as SEM [Bradley et al., 1999], Sato's algorithm [Sato, 1999] and Thiesson's incremental EM [Thiesson et al., 2001] may be used. One might also accelerate the process using binned data. The number of components starts at two, then increases until either it reaches the hard limit or the Bayesian information criterion worsens. For each number of components tried, the estimator runs multiple passes with a variety of initial states since a poor start may lead to convergence at an unacceptable local minimum.

Mixtures of independently, doubly-truncated normals are far harder and more expensive to estimate for a variety of reasons. Primarily out of curiosity and as an attempt at thoroughness this investigator made an attempt, which is described in more detail in Appendix B. An understanding of that method is not not necessary for following the rest of this thesis, and the estimator in practice never proved to be a factor except possibly consuming excessive CPU time – in contrast to the single doubly-truncated normal, which turned out to be the second-most frequent distribution among those which survived the goodness-of-fit criteria, behind the mixture of normals and ahead of the usual normal.

5. Pareto:

   The two-parameter form chosen happens to be well-known and inexpensive to estimate.

6. Uniform:

   The endpoints of the distribution are set just outside the range of observations.

A cumulative distribution function for a distribution that exactly models the data density nonlinearly scales that data to a uniform distribution within the range $[0, 1]$. The hope is that uniform distributions within uniform ranges helps find dependencies and models.

An obvious suggestion might be to use quantiles, if achieving uniformity is so helpful. This was considered but rejected as a scaling method for the following reasons:

1. Such a mapping still requires other forms of estimation or interpolation for values between observations.
2. Extrapolation to data outside the observed range is even harder; linear extrapolation, for instance, will do a poor job on data if the distribution is highly non-uniformly distributed per a lognormal or similar.
3. These mappings are highly parametric and cannot be considered summaries; nor are they as intuitive or readily interpretable as common distributions such as normals.

4.1.3 Integer variations

In recognition of the frequent appearance of all-integer data, the system determines whether the original unscaled attribute contains only integers. In such cases, instead of explicitly including a discrete version of each probability distribution, the scaling code uses wrappers to create new versions of the probability functions to reflect the shift in density

$$\forall x \in \mathcal{Z}\ g(x) = \int_{x}^{x+1} f(x)dx$$

and thus

$$\forall x \notin \mathcal{Z}\ g(x) = 0$$

The scaler then uses the Nelder-Mead simplex algorithm to adjust the parameter estimates to better reflect the discretized probability functions and the observations by maximizing log-likelihood [Nelder and Mead, 1965]. Other metrics might suggest other methods.

In summary, the axis scaling system examines attributes individually. For each attribute, it considers the full cross-product of two sets: a set of Box-Cox transformations, limited by possible values of λ; and a set of probability distributions, each with its own estimator and generator functions for the cumulative distribution, probability density, and percentage-point functions. It evaluates every triple of transformation, estimated fit and attribute according to Anderson-Darling A^2 score and quantile-quantile correlation. Surviving triples proceed to the next phase. In addition, the scaling system always preserves the original, unscaled data regardless of whether any scaled version survives the filtering criteria. For example, on a small data set containing `area` and `population` of 235 regions in the world, 12 different scaled versions of the `area` attribute and 13 different scaled versions of the `population` attribute survived in addition to the original versions; with the small size of the set. The logs include specifications for each attribute and its attendant scaling; thus

```
population-scaled-18: integer
        +original=population [1]
        +bc_lambda=0.333333333333333
        +bc_shift=0
        +d_name=gamma [DISCRETE]
        +d_params=0 1.45281863128665 376.542767985729
        +qq=0.995833093322018
        +A2=0.963965799859636
        +chi2=0.01
        +ll=-2559.12532341356
        +BIC=5141.88019766682
```

declares that scaled attribute `population-scaled-18` is an integer attribute based originally on `population`, with a $\frac{1}{3}$-power Box-Cox transformation with no shift, and after the application of the cumulative distribution function of a gamma distribution with location, shape and scale parameters of 0, 1.453, and 376.54 respectively. The quantile-quantile correlation, Anderson-Darling A^2, χ^2, log-likelihood and Bayesian information criterion scores all reflect specific goodness-of-fit tests; of these, Briareus uses the quantile-quantile correlation and the A^2 score, but leaves the rest for informational purposes. Figure 4.2 shows this particular function.

4.2 Model input selection

The axis scaling phase results in one or more versions of each original attribute, where each version corresponds to either an identity transformation or a pairing between a particular Box-Cox transformation and a probability-based scaling. The input selection phase decides which versions of which attributes serve as model inputs with which scalings, discards alternative versions of the input attributes, and labels the remainder as possible outputs. Models will later be built with the input attributes for

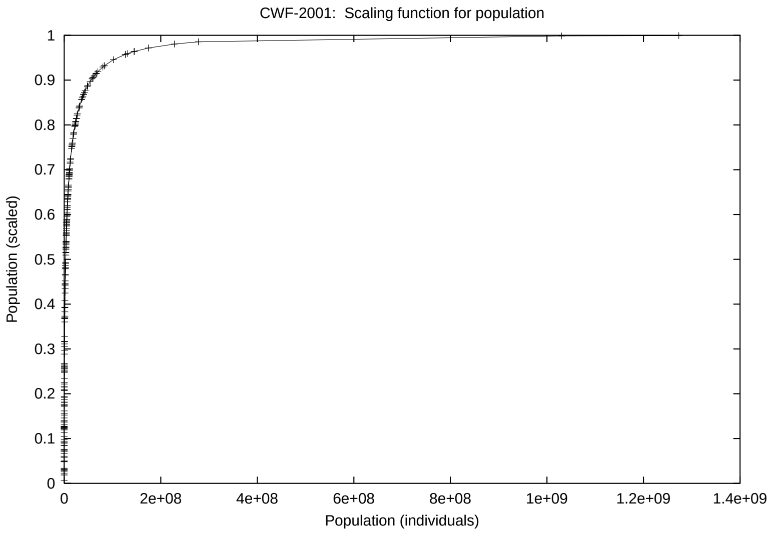


Figure 4.2: The axis scaling function applied to the `population` attribute of the `CIA World Factbook 2001` data. The original distribution, as indicated by the marks, is highly skewed towards low values; therefore, most of the scaled range is assigned to only a small part of the unscaled domain.

each of the possible outputs, to be selected among based on criteria to be described later.

Exhaustive searches of the space of possible labelings is clearly impractical, as the input-output assignment space grows exponentially with the number of attributes. Therefore, one needs to apply a more directed search. In addition, the selected model class of piecewise-linear regression trees is fairly expensive to evaluate; hence, this phase could use a faster heuristic to estimate the desirability of any particular input-output mapping.

Two significant issues, then, are

1. How to evaluate a set of input/output labels less expensively than building and evaluating the resulting models.
2. How to use this evaluation to search the solution space of input/output label assignments.

4.2.1 Evaluating an input/output assignment

For the first problem, Briareus employs the D_2 fractal dimension. Given distance metric d, let the D_2 value of data set X be

$$D_2(X) = \frac{\partial \log p_r\left(X\right)}{\partial \log r}$$

where

$$p_r\left(X\right) = \left|\left\{(x,y)\,|x,y \in X \wedge d\left(x,y\right) \leq r\right\}\right|$$

The above only strictly holds for perfectly self-similar infinite sets. For finite sets, one can use a variety of arbitrary criteria to determine a range of radii for which the relevant partial derivative appears sufficiently stable. This system examines all contiguous sets of half-plus-one of the $\langle \log r, \log p_r \rangle$ pairs, and computes the slope of the set with the lowest least-square error from the best-fit line.

Let us consider this in more concrete terms to convey an intuition as to the meaning of D_2. For a trivial example, suppose we have an infinite point set in which the points fall uniformly on a line in a 1-dimensional space: thus we might have

$$X = \{-\infty, ..., 1, 2, 3, ..., \infty\}$$

For a radius of $r = 1$, every point is within r of itself and two neighbors. For a radius of $r = 2$, every point is within r of itself and four other neighbors. Quadruple the radius to $r = 4$, and now every point has eight other qualifying neighbors. Every time the radius doubles, so does the number of "other neighbors" doubles. Hence, the $p_r \propto r$, $\log p_r \propto \log r$, and $D_2(X) = 1$. And the above would hold true regardless of the embedding dimensionality; it depended on the linear relationship, not the unidimensional nature of the overall space.

For a infinite plane X with uniform density, the number of points that qualify as other neighbors is proportional to the square of the radius. Then $p_r \propto r^2$, $\log p_r \propto 2 \log r$, and $D_2(X) = 2$ – regardless of how many dimensions are involved. Again, what matters is that the point set was planar.

For point sets with different properties, fractional D_2 values are possible. Thus, for the famous Sierpinski triangle set of Figure 4.3, the D_2 fractional dimension would ideally be $\frac{\log 3}{\log 2}$ if the set were infinite; Figure 4.3 only shows the first 3,280 points in the series. Figure 4.4 shows the corresponding graph of $\log p_r$ versus $\log r$, and a linear fit for most of those points with a slope of 1.6557.

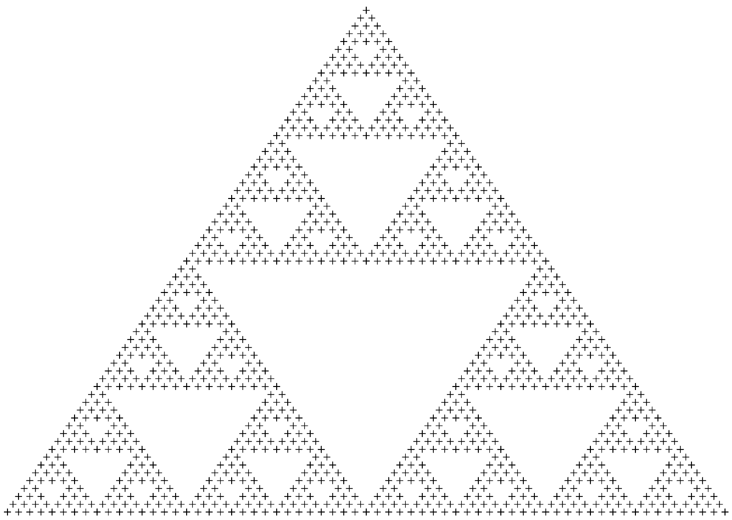


Figure 4.3: The Sierpinski triangle.

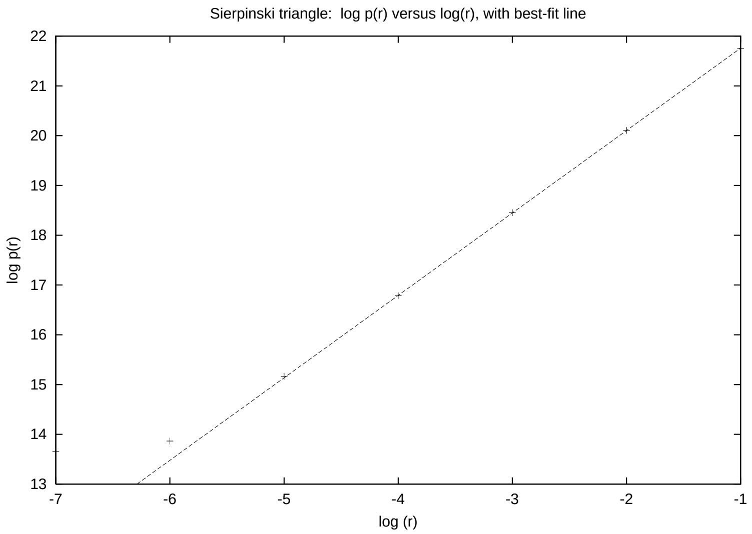


Figure 4.4: $\log r$ vs $\log p(r)$ for the Sierpinski triangle.

In general, for any given point set the more dimensions necessary to define the data, the higher the D_2 fractal dimension should be, regardless of the embedding dimensionality. Data defined by many independent attributes should have a high $\log p_r$ versus $\log r$; sets which can be captured by a few independent factors, such that they exist on lower-dimensional manifolds, should have a lower D_2 fractal dimension. A crucial caveat, however, *sans* scaling, one attribute can overwhelm another through a distribution with a spread of far higher magnitude. Take a square and stretch it too much in one dimension, and it resembles a lower-dimensional line; likewise, its D_2 fractal dimension will drop towards that of a line.

This relates to the attribute selection or labeling problem for the following reason: we can use it to assess how valuable a (possibly nonlinearly scaled) attribute is for an input. If we have a set of already-selected inputs, possibly empty, and we add another possible input, then the change in D_2 should reflect whether or not this attribute acts as a significantly independent factor that may help us define the space – or whether it instead adds little due to correlations with already-selected inputs. This hypothesis led to the methodology described in the following section.

4.2.2 Using the D_2 fractal dimension

Let us note that the D_2 fractal dimension may be estimated with a computational cost that varies linearly with the data set cardinality. The particular details may be found in Appendix C.

Let $\mathcal{X}$ be the data set, with set of attributes $\mathcal{A}$. The algorithm needs to choose the set of materialized attributes $\mathcal{M} \subseteq \mathcal{A}$. $\mathcal{M} = \emptyset$ is permissible if and only if all attributes contain no information due to being constant; the extreme of $\mathcal{M} = \mathcal{A}$ should only happen if all attributes have significant independent content.

FDR has one parameter, τ, which provides a hard threshold on the maximum acceptable per-attribute D_2 penalty.

Let $D_2(\mathcal{A})$ denote the D_2 fractal dimension of $\mathcal{X}$ projected onto $\mathcal{A}$. Then, FDR works as follows.

1. Initialize $\mathcal{M} = \mathcal{A}$, $D = D_2(A)$. $\mathcal{M}$ is the current choice of attributes to retain, while D is its fractal dimension.
2. Find a, D_a such that $D_a = \max_{a \in \mathcal{M}} D_2 \left(\mathcal{M} \setminus \{a\}\right)$. That is, a is the attribute whose removal which causes the least reduction or greatest gain in D_2.
3. If $D - D_a > \tau$, terminate and return $\mathcal{A}_M$.
4. If $D - D_a \leq \tau$, update $D \leftarrow D_a$ and $\mathcal{M} \leftarrow \mathcal{M} \setminus \{a\}$.
5. If $|\mathcal{M}| = 0$, terminate and return $\emptyset$; no attributes remain.
6. Otherwise, return to step 2.

Figure 4.5: The fractal dimension reduction (FDR) algorithm

That said, the question remains of how to use it. One possible approach is to use the fractal dimension reduction (FDR) algorithm previously developed by a group including this investigator for fast feature selection [Traina Jr. et al., 2000]. The FDR algorithm as per Figure 4.5 enumerates the steps. FDR performs greedy backwards-elimination; starting from the state where all attributes are to be retained, it iteratively and greedily removes attributes from that set until it sees an unacceptable reduction in the fractal dimension. When all remaining attributes cannot be removed from the present projection without such a reduction, the algorithm concludes that it has a set of attributes such that all have non-trivial independence from each other and should therefore be kept; the dropped attributes, on the other hand, have been dropped without significant loss and should be correlated with those that remain.

The problem dealt with by the FDR algorithm does differ significantly from that specified by this thesis, however. One of the more obvious and significant differences is that courtesy of the axis scaling phase, multiple versions of each original attribute may well exist. Thus, the embedding dimensionality and the redundancy may be far higher to start. This is problematic for backwards-elimination; while it may be tractable to as high as the sixteen attributes of one set [Traina Jr. et al., 2000], it is less clear that it would apply well to 233 different versions of 30 attributes as results from applying the axis scaling to the Dow Jones Industrials data set described in the next chapter as the fractal dimension be needed to be computed on projections involving many attributes. The first step would require the computation of the fractal dimension on 233 dimensions; then, there are 233 possible projections involving exactly 232 of those attributes, and so forth. With forwards selection, we can constrain the projection size by stipulating that there be at most one version of each original unscaled attribute in any projection under consideration.

Nor does the FDR algorithm explicitly prohibit choosing to keep two different versions of the same attribute; while it should be unlikely, it is theoretically possible. A selection approach, on the other hand, can enforce this outright by not allowing the selection of one when another has already been chosen. Finally, the work in [Traina Jr. et al., 2000] and the motivation for FDR treated the D_2 fractal dimension as one of the primary metrics. Here, the emphasis lies with modeling the relationships between attributes, and we prefer a bias towards selecting a few attributes and modeling the rest if possible.

Therefore, for the purposes of choosing inputs – including ensuring the selection of at most one version of each original attribute as an input – this phase uses the multistate hill-climbing forwards-selection approach shown in Figure 4.6. It's a greedy forwards-selection iterative search – hill-climbing – but rather tracking just the single best state, it maintains a priority queue of candidate states limited in size to the original number of unscaled attributes in order to reduce the probability of greedy selection converging on a poor local minimum.

In the case of the `CIA World Factbook 2001` data, the selector must choose its model inputs from the 27 different versions of the two original attributes. From the

1. Initialize $\mathcal{Q} = \{(\emptyset, 0)\}$, $\mathcal{Q}$ is a priority queue sorting combinations of attributes by fractal dimension in descending order. In order to limit run time, it will hold at most $|\mathcal{A}|$ combination-fractal dimension pairs.

2. Initialize $\mathcal{T} = \emptyset$. $\mathcal{T}$ is a table that tracks which combinations have been explored.

3. Initialize $\mathcal{M}$ as the best solution found so far, $\emptyset$, with fractal dimension $D = 0$.

4. Let $\mathcal{O} : a \rightarrow b$ be the relationship that maps attributes a to unscaled version b, and permit the obvious extension for sets.

5. Until $\mathcal{Q}$ is empty:

   (a) Let D' be the highest fractal dimension currently in $\mathcal{Q}$.

   (b) Select and dequeue a pair $(\mathcal{S}, D')$. Decide ties arbitrarily.

   (c) Let $\mathcal{X}$ be the set of possible "next" states;
   $\mathcal{X} = \{\mathcal{S} \cup b \| \, (b \in \mathcal{A}) \wedge (\mathcal{O}\,\{(b)\} \notin \mathcal{O}\,\{\mathcal{S}\})\}$.

   (d) Filter $\mathcal{X}$ so that $\mathcal{X} = \mathcal{X} \setminus (\mathcal{X} \cap \mathcal{T})$.

   (e) For each combination $\mathcal{C}$ in $\mathcal{X}$, compute the fractal dimension D_C and add $(\mathcal{C}, D_C)$ to $\mathcal{Q}$. Ignore if gain is lower than 0.7; this is a rather high threshold that could be lowered if one wants more accurate models, but fewer of them and dependent on more inputs.

   (f) If $|\mathcal{Q}| > |\mathcal{A}|$, set $\mathcal{Q}$ to contain only the $|\mathcal{A}|$ best entries according to the already-computed D_2 values. Break ties arbitrarily.

6. Use the best-known solution $\mathcal{M}$ with fractal dimension D.

Figure 4.6: Multiple-state hill climbing.

initial empty state, it considers the fractal dimensions of each individual version; the top two choices are a previously mentioned version of `population`, scaled with a $\frac{1}{3}$-power Box-Cox transformation and a gamma cumulative distribution function (CDF) with a fractal dimension of 0.9197; and a version of `area`, scaled with a logarithmic transformation and the CDF of a three-component normal for a fractal dimension of 0.9137. It proceeds by examining every pair of the former with each scaled version of the `area` attribute, rejecting every pair due to their fractal dimensions not being higher than $0.9197 + 0.7 = 1.6197$. The same process occurs for the successor states of the selection of that particular scaled form of `area`, again yielding no pairs meeting the threshold. The search terminates with just that particular scaled form of `population` having been selected; the other versions for `population` are ignored, and each of the 13 different versions of `area` are possible model outputs.

It may be noted that while this algorithm restricts the final set of attributes to only one version per attribute, nothing prohibits the use of a model class which itself uses multiple versions of the attributes it receives. For example, an algorithm for estimating polynomials might involve multiple powers of the same attribute. It should also be stated that while in theory the "explored-state" table $\mathcal{T}$ could grow excessively large should the multistate hill-climbing explore too many states, this does not seem to have been an issue in practice.

Both the original FDR algorithm and the priority queue-driven version have non-trivial drawbacks. Three of the most obvious issues originate with the use of the D_2 fractal dimension. First, categorical data poses problems. For the original definition of D_2 to apply, one would need a heterogeneous distance function such as those found in [Wilson and Martinez, 1997]. Furthermore, the "nested grid" family of fractal dimension computation methods would no longer be relevant, barring a full symbol hierarchy. The design chosen bypasses this problem by always materializing nominal attributes and thus never involving them in the selection phase.

Second, both algorithms may incur substantial computational cost. Every iteration requires D_2 computations on numerous different projections, and these D_2 computations will be more expensive the more attributes are involved in the projection.

Third and perhaps most serious, D_2 lacks clear definition when the current projection results in a non-self-similar data set. This case can be detected through measuring the linearity, or lack thereof, in the point set defined by the estimated $\log p(r)$ and $\log r$ values. Unfortunately, detection of such a condition does not imply the existence of a solution. This system currently ignores the offending projection and notes this occurrence. A quick `grep` through logs shows that of 22,313 fractal dimension computations involved in the attribute selection over the original and axis-scaled versions of nine data sets, 1,799 estimates were rejected due to having too low correlation.

A different issue arises with the separation of modeling from attribute selection. The system attempts to select attributes by independence, but this is an incomplete picture; two attributes may be very dependent on each other and similarly uniform on their respective projections, but one attribute may be a far superior choice to the other for the purposes of function fitting. Such a pathological case can be seen in Figure 4.7. To completely avoid this without relying on human interaction, it may be necessary to incorporate some form of model construction at this stage – perhaps on a small sample, so as to avoid excessive computational cost.

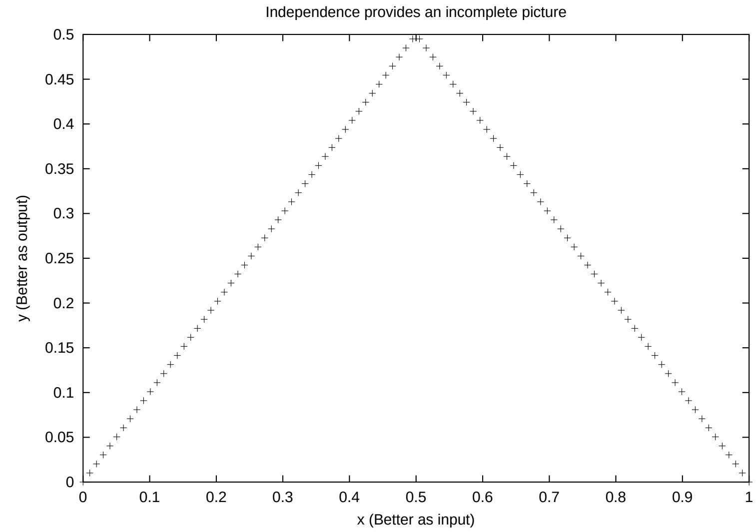


Figure 4.7: A pathological case for the attribute selector. The projections along the two attributes are equally uniform, but x makes far more sense as an input than y.

4.3 Modeling

The final stage performs modeling. As per the previous section, we may divide the set of attributes into three sets: the specific scaled versions of attributes selected for materialization, the non-selected other versions of attributes selected, and all versions of attributes which have not been materialized in any form. The first set provides the possible inputs for the models; the second set may be ignored; and the third set defines what attributes Briareus will model, although for each underlying attribute only the best model need be selected – for any given definition of 'best'.

urrently, the modeling phase uses regression trees with linear or constant models in the leaves as implemented by the RT system [Torgo, 1999]. Little actually constrains the choice of modeling system here, other than the need to work on continuous attributes and to be usable with minimal human intervention. For instance, it would not be reasonable for the modeling system to be frequently querying the user about how many hidden units should be used by a neural network or how they should be connected. Prior versions used a research version of RT due to its flexibility, availability and capacity for automation [Torgo, 1999]; discontent with the limited pruning coupled with unavailability of the source code led this investigator to implement a regression tree system based on SECRET [Dobra and Gehrke, 2002] but with a much more aggressive pruning system that considers complexity in addition to accuracy. To further reduce overfitting, a form of four-fold cross-validation is performed. Details about the major differences between this tree and SECRET, and the cross-validation procedure, may be found in Appendix D.

In most sets, at least one attribute will not be selected as an input in any given axis scaling. For each such attribute, the modeler must select a model targeted at one particular axis scaling among those that survived the goodness-of-fit checks at the end of the axis scaling phase. The approach taken consists of iterating through the possibilities. For each candidate version, the set of possible inputs consists of the specific transformed attributes that were chosen. There are two main categories of metrics which traditionally influence this sort of selection: complexity and accuracy.

4.3.1 Complexity / compression

As described in Chapter 3, there exists a substantial body of work devoted towards considering the complexity of models as a method for choosing either within or among model classes. These methods assess costs for storing the model, such as its structure and parameters; in addition, they may either operate solely on discrete output and require perfection, thus eliminating any need to directly add accuracy and storage costs in a single metric, or they may permit inaccuracy at a price, such as by assessing the number of bits used for storing enough information to correct an error to within a demanded error tolerance.

Methods which cover model classes that accept input data, such as the attributes that have been labeled inputs in this domain, also should be accounted for when dealing with cases in which this cost may change. For instance, with Briareus there is no guarantee that the number of attributes labeled as inputs will not differ between the scaled and unscaled cases.

The experimental results in the next section do partly describe the complexity of the model in terms of how many nodes exist in the regression trees, as this can be readily quantified. However, there are numerous different ways one may decide to assess the complexity of scaling methods – arithmetic operations, cycles required, lines or bytes of code in a given language and implementation – and none is canonical. Likewise, to use a method that deals in bit costs requires greater care with floating-point numbers.

This thesis focuses on the values within the data, not the bits used to represent the values. Therefore, while tree sizes are reported for comparison purposes in terms of the number of nodes, it does not suggest any particular encoding or balance between accuracy and complexity as canonical, or attempt to provide sufficient different such methods as to seem comprehensive. Their relative merit will depend upon the application.

That said, for the few attributes used which happen to be completely integers, numbers are provided which correspond to bit costs involving standard prefix-free codes for the integers, and IEEE 64-bit doubles for floating-point numbers among model parameters. These numbers stem from tree selection influenced by this metric, and do not necessarily relate to performance of the trees selected by the metric of the next subsection. They are also presented only for the curious, rather than as results to be significantly analyzed.

4.3.2 Accuracy

The evaluation therefore rests primarily on accuracy. Fundamentally, this is the problem of deciding "how good" a model M and scaling function S when dealing with input tuple $\hat{x}$ and output datum y. This is more complicated than it seems.

Scaling: For instance, many error metrics might give different rankings among model classes depending on whether one evaluates in the original space

$$S^{-1}(M(\hat{x})) \sim y$$

or scaled space

$$M(\hat{x}) \sim S(y)$$

due to the nonlinear nature of S. Figures 4.8 and 4.9 present such a pair; they both describe the same data and the same model with the only difference being that the latter presents the results in the same scaled space in which the model was generated. Intuition suggests that since both cases cover the same model on the same data, differing only by a reversible transformation, that they should be considered equally good. Barring an assumption about the goodness of either the original scaling or the method under evaluation, it seems best to specifically devise a method which is reasonably scaling-invariant.

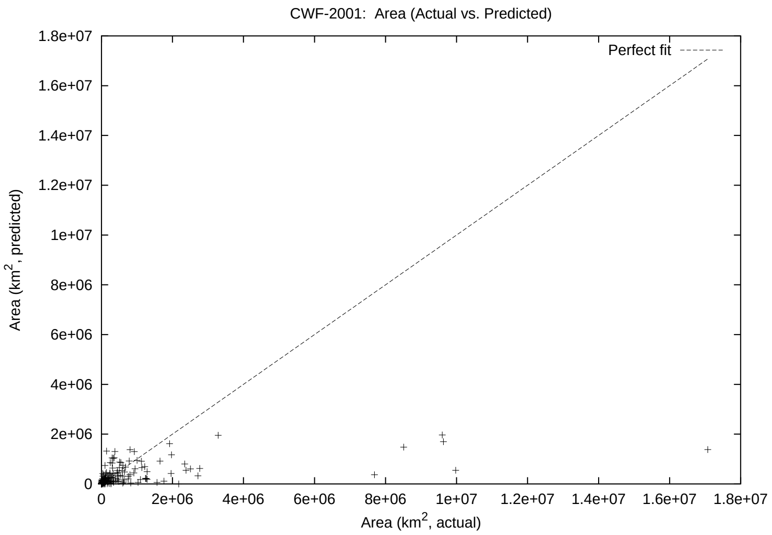


Figure 4.8: From the CIA World Factbook 2001, actual versus predicted area in the original space but based on a model generated in a scaled space.

Carelessness regarding this constraint can lead to rewarding extremely perverse data if evaluating in the scaled space. For instance, methods based on absolute error could be tweaked by scaling an axis such that most of the correct outputs end up with low-magnitude values, while relative error metrics suggest arbitrarily shifting the data by adding a ludicrously huge constant to every desired output and prediction such that the errors get reduced by the much higher denominators.

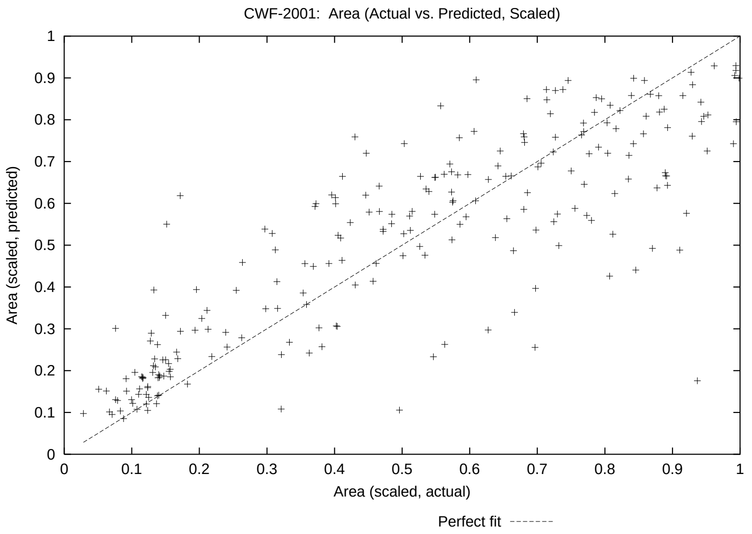


Figure 4.9: From the CIA World Factbook 2001, actual versus predicted area in the scaled space where the relevant model was built.

Reals: The metric must work on real numbers, not just integers. While integer-focused metrics may be usable after discretization, actually choosing and justifying any given discretization scheme is non-trivial in the absence of specific application constraints and domain knowledge.

Resolution: For any given attribute, the distribution of the observations therein might be of highly variable density. It seems unreasonable to propose that error tolerances be tighter in areas of high density, and can be more relaxed in areas of sparser density, as the same error in absolute terms may cause more confusion in a dense region than a sparse region.

Deriving the quantile metric Thus, the investigator designed a quantile-based error metric rather than more well-known metrics such as the sum-of-squared error and relative-error variants thereof. For instance, SPARTAN [Babu et al., 2001] relies on the data – inputs and outputs – being discrete, and therefore can penalize accuracy by assessing the bit cost of storing the integer corrections. There are a number of ways to extend this to reals involving discretization, but the schemes may give different results – and in addition it leaves unanswered such questions as why linear shifts will affect costs unevenly at different values in the data range, and whether to test in the original or scaled spaces. If one adds a constant that is several trillion times the maximum observed value to all outputs and predictions, for instance, their relative bit costs will become much closer to each other regardless of accuracy.

For the output attribute $\hat{y} = (y_1, ..., y_n)^T$, input matrix $X = (\hat{x}_1, ..., \hat{x}_n)^T$ with $\hat{x}_i$ the k-dimensional input vector for output y_i, scaling function $S : \mathcal{R} \to \mathcal{R}$, and modeler $M : \mathcal{R}^k \to \mathcal{R}$, the quantile error metric is computed as follows.

1. Define mapping $Q : \mathcal{R} \to \mathcal{R}^2$ as follows:

   Map every y_i value to the range defined by the minimum and maximum observed rank of y_i. For instance, if the highest 10% of the $\hat{y}$ attribute all consists of the same value v, but the next-highest value differs, then for all i such that $y_i = v$, the quantile range $Q(y_i) = [0.9, 1]$. Values which only occur once in $\hat{y}$ will have the same value defining both extrema of the "range".

   For values not in y_i, let either Q linearly interpolate based on the neighbors on either side, or linearly extrapolate based on the relevant extrema and second most-extreme point. In both cases, it should map to ranges containing only a single value.

2. For assessing the goodness of any one prediction, examine $Q(y_i)$ versus $Q\left(S^{-1}\left(M\left(\hat{x}_i\right)\right)\right)$. Use the absolute magnitude of the closest distance between the ranges.

   If the quantile ranges intersect, the distance is 0; otherwise, use the lesser of the distances between the maximum quantile of one and the minimum quantile of the other.

3. For assessing the combination of M and S, sum the squared errors in Q space.

Figure 4.10: Quantile error metric.

Figure 4.10 gives the procedure for computing the quantile error. Note that the procedure transforms data into a space defined by Q, which is based on the observed quantiles. This has the effect of stretching dense regions and compressing sparse regions, since this is based on ranks more than absolute value. If, for instance, the maximum observation were increased arbitrarily, it would still be mapped to 1 and the next-highest would still be mapped to $\frac{n-1}{n}$. In fact, the only mappings that would change would be for values exceeding the second-highest observation, and the effect would be limited for any value that was at or below the original observed maximum. Likewise, linear shifts do not affect results so long as the same shift applies to both the predictions and the observations.

Even the choice of scaling S has a limited impact so long as it is order-preserving. For values in the original data set, S has no effect; for unscaled values not in the original data set, the need to interpolate to estimate their ranks based upon the closest data means S affects the results slightly.

Using the quantile metric Computing errors in Q-space lets us present multiple types of results. First, to assess the overall combination of input selection, axis scalings (for both inputs and outputs), and model, we can present the sum-of-squared Q errors.

Second, to give a more detailed instance, median quantile errors are also frequently provided. These can be readily interpreted in terms of rank; for instance, a median quantile error of 0.05 means that half the time, the errors were no more than 5% of the data set off – in terms of rank, not absolute value. A trivial baseline is that a constant model which always predicts the median observation should have a median quantile error of 0.25, or less if there are duplicates.

Third, for yet more detail graphs can be generated which show rank versus quantile error for a single model so that their distribution is clearer. For a good model one may expect most of the quantile errors to be low followed by a sharply increasing tail encompassing major exceptions to the rule.

Fourth, individual quantile errors can help flag tuples that have much higher errors than their brethren. For instance, recall that applying nonlinear axis scaling and the input selector to the two-attribute `CIA World Factbook 2001` data set yields the selection of a particular scaled form of `population` as an input, and thirteen versions of `area` as possible outputs. The most accurate of these models is a single-node tree of the form

$$\text{area\_km2-scaled-3} \leftarrow 7.926 \times 10^{-2} + 0.8508 \times \text{population-scaled-18}$$

with the data domain files providing sufficient information to interpret the specified scalings. If one computes the individual quantile errors for all 235 tuples, a striking gap emerges in that the highest quantile error has a magnitude of 0.7121, but all other quantile errors are 0.4560 or below. It is therefore obvious that one particular tuple is being modeled extremely poorly compared to the rest.

Quantile transformation before modeling? Given the use of the quantile transformation for evaluation, an obvious question is whether the models themselves should be built using the quantile transformation. One could consider applying this to all the attributes instead of the aforementioned nonlinear scaling phase; alternately, it could be applied to only the outputs.

From the perspective of minimizing eventual quantile error, this may make sense. It should be noted, however, that quantile-based scaling has its own issues, such as a complete lack of interpretability, difficulty extrapolating when working with potentially open-ended distributions, and the need to do *de facto* density estimation for interpolation. The interpolation and extrapolation issues have more potential impact when building a model that may later be applied to new data, especially outside the previously observed ranges.

4.4 Comparison with SPARTAN – Design

At this point, it seems relevant to compare and contrast with SPARTAN [Babu et al., 2001]. First, SPARTAN does not use any form of scaling on the data, instead working on it as-is. Thus, the number of attributes does not change, and no issues arise with multiple versions of the same original attribute. Second, SPARTAN follows the "wrapper" design concept instead of the "filter" concept; instead of using a separate heuristic, SPARTAN employs the actual modeler – CART-based regression trees with constant values in the leaves – when evaluating which attributes do not need to be stored as inputs. To reduce the computational cost, SPARTAN samples the data. SPARTAN also discretizes the data, then applies a Bayes network construction algorithm in order to identify dependencies and make an initial determination of which attributes would be useful for modeling which attributes. SPARTAN may alter the choice of inputs in later iterations to reflect which attributes have previously been chosen to be modeled, but the Bayes net provides the start configuration.

In theory, SPARTAN's wrapper design should provide superior models compared to a three-phase, acyclic design. On the other hand, its lack of scaling combined with piecewise constant models versus nonlinear scaling and piecewise linear models may make the models less efficient in the presence of sufficiently complex dependencies, and SPARTAN only applies to discrete data. In addition, the CART trees do not generalize to output values beyond that of the original set; it does not interpolate or extrapolate.

One obvious question comes to mind: Could the scaling phase proposed here simply be prepended to a version of SPARTAN rebuilt around piecewise-linear regression trees, and modified to take into account transformation parameter costs? Without a substantial reorganization of SPARTAN, the answer is no; SPARTAN relies heavily on Bayesian networks and the discrete nature of the data, whereas Briareus concerns itself with attributes on continuous domains. In addition, even if a discretization method were specified, SPARTAN would need to be modified to deal intelligently with multiple versions of each original attribute. Also, if the data were discretized to enable SPARTAN-style attribute selection, one would have to decide whether to retain piecewise-constant trees, or to use the non-discretized data to build piecewise-linear trees while knowing that the Bayesian-based selector does not even consider the order of values for each attribute let alone the relative magnitude of the gaps between them. Thus the domain difference results in systems that do not appear to be well-suited for straightforward combination or perhaps even direct comparison.

Finally, the systems use different metrics of success. SPARTAN focuses on compression, and therefore uses error metrics that translate errors into bit costs; Briareus treats the modeling accuracy as more important, and endeavors to use a metric that takes into account local data density in exchange for ignoring compression costs. While SPARTAN with scaling enabled would favor certain perverse scaling methods that would provide extremely dense regions, the quantile-error metric was designed to not reward this.

4.5 Scalability

As with any complex system, it would be of interest to know just how long execution might take, and whether Briareus would scale to large data sets either as-is or with moderate modification. One might want to know what a worst-case scenario is. Briareus appears to be sufficiently complex that an empirical analysis would be more tenable, however.

4.5.1 Numerical methods

The scaling methods used frequently involve the solution of problems for which no acceptable closed-form approximation exists. For instance, the cumulative distribution function of a single univariate Gaussian involves integration over $(-\infty, x]$ for some $x \in \mathcal{R}$. This is not amenable to a perfect closed-form solution. Iterative numerical integration methods exist, such as the well-known Romberg rule and Simpson's method, but these have two major issues in that execution time is related to arbitrary convergence criteria and that the theoretical guarantees of correctness and convergence of other algorithms may not hold when given approximate values.

The worst case is that convergence which should occur, does not, and that existing checks for this behavior do not suffice. In that theoretical case the worst-case complexity would be infinite. The existing checks aimed at avoiding this do cause certain failures; for instance, the approximations in the current implementation have failed on computing the percentage-point function of a Gaussian distribution when the percentile in question is very close to either extreme. That latter computation involves not only numerical integration, but also numerical searching to invert the integration-based cumulative distribution function for specific values.

The same holds at a higher level, such as for certain distributions. Expectation-maximization is used to estimate univariate mixtures of normals. Barring issues related to numerical approximations causing non-ideal behavior, `EM` will converge to local minima – but the guaranteed bounds are loose. For any given data set, convergence could occur much more rapidly depending on the initial state and the actual density of the data; thus, even if Briareus recorded the times spent on scaling, extrapolation to other sets would be unreliable. This method could, however, be replaced with first taking time linear in the number of tuples to generate a constant-sized histogram, followed by constant-time estimation.

The construction of SECRET trees involves the estimation of binary mixtures of multivariate normals [Dobra and Gehrke, 2002]. The current implementation uses SEM [Bradley et al., 1999] bootstrapped by a k-means clustering algorithm; while SEM scales reasonably well after initialization due to its use of summaries, k-means is less predictable. If all or almost all of the data would be associated with a single cluster by k-means, it may take a considerable number of iterations to discover the second cluster, if any. Different initialization methods might be more predictable.

Based on the timing experiments presented in Section 5.4 of the next chapter, the modeling phase – which includes the construction, pruning, selection, and final evaluation of the SECRET trees – completely dominates computational cost with the current implementation.

4.5.2 Inability to predict the number of retained inputs

The number of transformed attributes that survive filtering depends on the number of transformations allowed, as well as how close each particular attribute is to any given distribution and how one tweaks the estimation of each distribution and the goodness-of-fit criteria. It is possible that minor tweaks would have a significant impact here even with the basic algorithms otherwise unchanged.

The number of transformed attributes greatly affects the potential worst-case complexity of the scaling system, as more attributes increases the space it has to search. However, if the attributes are highly correlated, then the search may terminate early as the incremental fractal dimension changes drop below the built-in arbitrary threshold. Hence, one needs to know more than the number of attributes here; yet, in the general case this number cannot be predicted merely by knowing the dimensions of the data set.

Likewise, the number of attributes selected as inputs affects the complexity of the generation of each model in multiple ways. More inputs mean larger sets for regression, but it may also make it easier to construct a sufficiently accurate model with fewer pieces; however, more inputs means fewer outputs and therefore fewer models to construct.

4.5.3 Complex tree construction process

If one assumes the successful termination of computation of split points, including the aforementioned use of the Scalable Expectation Maximization (SEM) algorithm [Bradley et al., 1999] for binary multivariate normal estimation to generate pseudo-class labels, and likewise for the estimation of linear models via either the singular value decomposition or an approximation based on recursive least-squares; then the tree construction will terminate as every node split requires that the child nodes each relate to strictly fewer data than the parent. The maximum number of nodes even without other checks such as stipulating minimum number of data to even consider a split would therefore be twice the number of tuples, less one – every split resulting in a child for a single-tuple leaf and a child for the rest of the data.

The above assumptions, however, do not shed much light on either worst or average case stability. Even for execution on a single data set, holding constant the size of the data and the number of inputs versus outputs, the observed time for generation of a tree might conceivably vary considerably due to variations in the number of splits required and therefore the number of other computations.

4.5.4 Estimates

With the above in mind, one can suggest at least some guidelines for the individual stages.

First, consider the nonlinear axis scaling stage. As all the estimators are applied independently to each attribute, the nonlinear axis scaling phase should scale linearly with the number of attributes m – if all the attributes are similarly expensive to process. If all the supplied distribution estimators are enabled, by far the most computationally expensive should be the estimators for the univariate mixture of normals, and for the univariate mixture of truncated normals. The former currently relies on standard expectation-maximization and presumably converges at the same rate, but could be modified to operate on binned data for essentially performance linear in the number of tuples n. The latter, as implemented, could in theory require at least time cubic in the number of tuples if the breakpoint divisions proved to be completely pathological, but this worst case may actually not be attainable. It would be worth considering the removal of this last estimator for sets of significant size, or refining it for improved scalability.

Second, we can examine the attribute scaling phase. Each invocation of the fractal dimension code in practice scales linearly with both the number of tuples and the number of attributes attributes. For an older version based on a pure single-state forward-selection search, without axis scaling, one could overestimate the computational cost in the worst case based on the number of tuples n, number of attributes m, and number of attributes eventually selected as inputs k as $\mathrm{O}(mnk^2)$ – the number of iterations is k, and the final iteration requires computing fractal dimensions on $m - k + 1$ projections of dimensionality k each. The actual algorithm used is less predictable, since no projection needs to be computed using two or more different scalings for the same attribute, and the system tracks more than one best state. Thus, if T is the number of possible axis scaling functions per attribute, we might very loosely estimate the complexity as $\mathrm{O}(Tmnk^2)$.

Third, we have the modeling phase. One can at least note that the number of models which need to be built scales linearly with the number of outputs – $m - k$ in the unscaled case, something more complicated in the scaled case – and that the dimensionality that these models need to be built in scales linearly with the number of inputs. With reasonably efficient algorithms, one might expect the model construction time to vary approximately linearly with the number of tuples; thus, $\mathrm{O}(T(m-k)kn)$ may be plausible. It is left to the experiments to check this conjecture.

It may be noted that regardless of the difficulty of assessing the scalability through theoretical analysis, the experiments documented in Chapter 5, Section 5.4 suggest that each section incurs a computational cost that scales linearly with the number of tuples. Judging from those same experiments, it is the final $\mathrm{O}(T(m-k)kn)$ modeling time that dominates, although if $k \sim m$ the $\mathrm{O}(Tmnk^2)$ of the input selection would actually scale worse.

Chapter 5

Experiments

To attempt a theoretical proof regarding the performance of such a collection of approximations and heuristics with respect to a problem with few assumptions would seem foolhardy. Hence, experimental testing seems appropriate.

5.1 Implementation

The system exists primarily in Perl 5 scripts and modules as well as some C++ code. Including internal documentation but excluding interpreters, other libraries and other pre-existing code, this consists of over 50,000 lines of code between the statistics software, the fractal dimension package, attribute selector, the modeling system, and assorted test harnesses. While numerous components have parameters that can be tweaked, for the purposes these experiments all such parameters were left at their general-purpose defaults – no per-set tuning was performed.

Testing proceeded on a single machine with two Intel Xeon 2.8 GHz processors and 1 GB of memory running Linux. With the relatively small sets used, processing power was far more important than memory. No attempt was made to exploit parallel computing.

5.2 Usage

As noted, while there exist numerous opportunities for adjusting parameters to best suit each data set none were taken. For any given data set, the procedure was as follows.

1. Write a "domain" file that labeled each attribute as `continuous`, `nominal` (symbolic), `integer`, or in a very few cases `ignore`. Certain attributes which were naturally outputs based on the set's purpose were labeled thus, as noted in the sections to come.

2. Run the axis scaling system on the data file.

3. Run the model-input selection system on both the unscaled and scaled versions.

4. Run the model-building and selection system for both cases.

5. Use additional scripts to compute quantile error for each tuple and associate them with the line numbers in the data set for reference.

6. Check the logs for aggregate error statistics.

7. Check the tree files for the number of nodes in each.

8. Start the outlier search by examining the tuples with the highest quantile errors, especially when these errors are much higher than the others or are within the same general time period where applicable.

5.3 The impact of nonlinear scaling

Earlier sections of this work contend that nonlinear scaling should be capable of having a significant positive impact on the accuracy and efficiency of the models. With a piecewise linear model such as those fit by RT and SECRET, nonlinear scaling might enable higher accuracy, greater efficiency, or both.

Regardless of one's expectations, it seems relevant to examine at least the following potential areas of observation.

Q1. What is the impact of nonlinear scaling on the attribute selection process? The implemented nonlinear axis scaling scheme has a definite bias towards increasing uniformity through its goodness-of-fit tests. The attribute selector, using the D_2 fractal dimension, prefers uniform and independent attributes. It therefore reasonable to suspect that more attributes will be selected as inputs after the application of the nonlinear scaling.

Whether or not this happens is certainly important, given that more selected attributes require more bits to store but may also be enable better models. Where one draws the line between storage cost and model accuracy depends on the application and must be left to the user.

Q2. How good are the resulting models? As suggested in the chapter on system design, two different mindsets – compression and modeling – suggest two different methods for assessing goodness.

With integer data, the traditional MDL or MDL-like approaches can be used to combine model complexity and model accuracy into a single metric measured in bits. The complexity of a model can be assessed in terms of the bits necessary for encoding the structure and parameters given a specific encoding scheme, and one can also devise schemes for assessing accuracy according to the cost of storing correction data. Such a model was described in the previous chapter; results using it are reported for the few cases of modeled integer attributes. These values should not be considered without remembering the problems inherent in such a model, such as the implicit assumption of the correctness of the original scaling. In addition, these bit costs do not apply well to floating-point outputs. Therefore, with both integer and floating-point data models the experiments also perform model evaluation and selection according to the quantile-based error method.

The quantile method does not take into account model size, so it also seems reasonable to report the number of nodes in the selected regression trees.

Q3. What else can we find out? For instance, should the sum of quantile errors for a particular model seem unreasonably high, one might examine the distribution. If most errors are in fact quite low, while a few errors far exceed the rest, we may have identified what we might term a "semantic outlier"; while not necessarily an outlier in terms of traditional metrics such as distance from neighbors, such a tuple represents a combination of values whose relationship has not been properly captured and may therefore be of unusual interest. In other cases, simple scalings or tiny but accurate regression trees may exist that provide useful information about the data.

The following experiments have been run using data sets involving factors not known to the experimenter. A variety of data sets from different domains has been used in order to reduce the likelihood of deriving conclusions that rely on unusual traits or coincidences. This section describes the results.

Some experiments have also been run with the same data sets, but restricted to strictly linear models instead of piecewise-linear models. These experiments are described in Appendix E.

5.3.1 Abalone

The Abalone set [Nash et al., 1994] hosted by the UCI Repository [Murphy and Aha, 1994] contains 4,177 9-dimensional tuples regarding the physical characteristics of abalones. The dimensions include a nominal attribute, `sex`, which can be either male, female or infant; multiple continuous attributes, namely `length`, `diameter`, `height`, `whole weight`, `shucked weight`, `viscera weight`, and `shell weight`; and one integer attribute, `rings`. The task originally associated with the set was to predict the number of rings; hence, this attribute was flagged as an output.

Without nonlinear axis scaling Without using nonlinear scaling, the system chose to retain three attributes as inputs – `sex`, `length` and `height`. Median quantile errors for these attributes ranged from 2.3946×10^{-4} for `rings` and 1.6705×10^{-2} for `diameter`, to 4.1562×10^{-2} for `shucked weight`. For reference, the mean squared error on `rings` was 6.9871. As only `rings` is an integer attribute, the other attributes lack bit cost results.

Attribute	**Bit cost**	$\sum$ **Q error**^{2}	**Nodes**
`Diameter`	N/A	8.8500	1
`Whole weight`	N/A	16.675	7
`Shucked weight`	N/A	30.808	5
`Viscera weight`	N/A	28.905	7
`Shell weight`	N/A	26.433	7
`Rings`	20,391	91.129	1

Table 5.1: Errors for the `abalone` set, without nonlinear scaling. Bit costs reflect the tree cost as well as errors, while quantiles do not.

With nonlinear axis scaling With nonlinear scaling enabled, three attributes were retained – `sex`; `shucked weight`, with a truncated normal; and `viscera weight`, again scaled with a truncated normal.

Regarding the output attribute, the 3-node tree for an untransformed `ring` attribute accrued a total of 93.720 in squared quantile errors, delivering somewhat similar results. The mean sum-of-squared error in the original, unscaled space was 6.8078; if we label all attributes as inputs, and use the accuracy-focused pruner instead of the balanced pruner, the resulting 25-node tree has a mean squared error of 4.361 rings, a sum-of-squared quantile error of 53.55, a median quantile error of 2.3946×10^{-4}, perfect prediction on 1,041 cases.

Table 5.2 summarizes the quantitative scores for the models in the case of nonlinear scaling. The quantile scores may seem worse; however, median quantile errors in both cases are quite small. For instance, the median quantile error for `shell weight` increases from 3.5026×10^{-2} in the unscaled case to 4.5170×10^{-2} in the scaled case; the maximum quantile error, however, has dropped from 0.76991 to 0.74054, and in the scaled case the maximum is substantially above the second-worst of 0.51826. In the case of `diameter`, with the worst relative penalty, the median quantile error still remains 2.9571×10^{-2}. In no attribute does the median quantile error exceed 4.8428×10^{-2} in the scaled case – this worst case occurring with `height` and the median quantile error did drop from 2.9680×10^{-2} to 1.8352×10^{-2} for `whole weight`. Recall that a median quantile error of 5×10^{-2} means that the predicted output would have a rank error of less than 5% half the time.

It should be noted that the six models in the general unscaled case require a total of 28 nodes in their trees, with a total sum-of-squared quantile error of 202.8, while the six models after nonlinear scaling take 20 nodes to deliver a total sum-of-squared quantile error of 232.4. If one considers the models post-scaling to be sufficiently good, then the size benefit is obvious and substantial.

Distribution of errors One who prefers to look at the error results in more detail might end up generating per-attribute percentage-point function curves such as those in Figure 5.1. Figure 5.1 shows that errors tend to be smaller in the scaled case than in the unscaled case. For a reasonably accurate model, we expect a smooth curve where most errors are low, and with a sharp rise in the magnitude of errors for the few exceptions to the rule. In such graphs, if the scaled and unscaled curves mostly coincide, they deliver similar performance; a broad divergence, on the other hand, indicates that the model behind the lower curve has a much better distribution of errors. Figure 5.2 shows only the worst-modeled cases from Figure 5.1; in both the scaled and unscaled cases there are a few exceptionally badly modeled tuples.

5.3 shows the corresponding results for the `rings` attribute; as one might expect for a discrete attribute, these percentage-point functions are step functions.

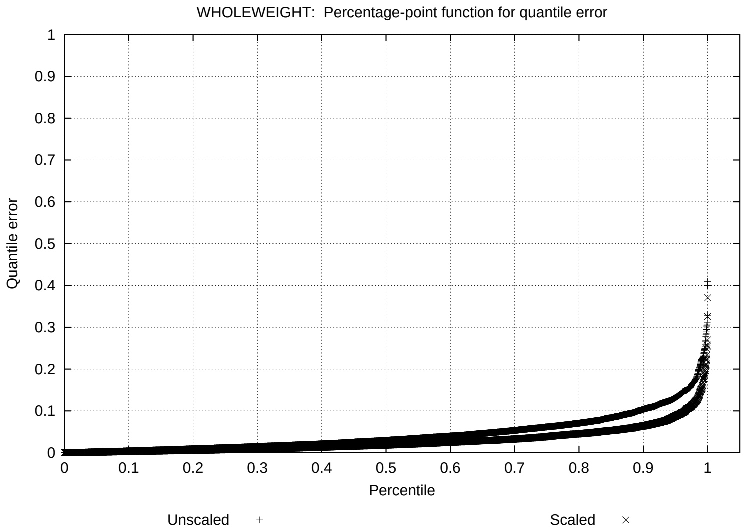


Figure 5.1: Percentage-point curves for quantile error in the unscaled and scaled cases for the `whole weight` attribute of the `abalone` set. A percentile of 50% means the median quantile error; 100%, the maximum. While the curve for the scaled case lies below the unscaled case, the differences are minimal except among the more serious errors. In both cases, the worst errors – percentiles 95% and up, for instance – are much worse than most errors.

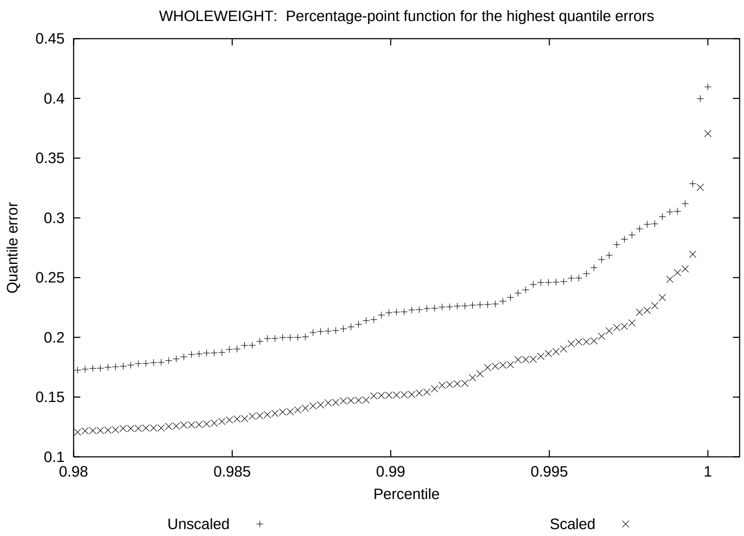


Figure 5.2: Percentage-point curves for the very worst quantile errors in the unscaled and scaled cases for the `whole weight` attribute of the `abalone` set. In both the unscaled and scaled cases, the worst errors are quite high relative to the rest.

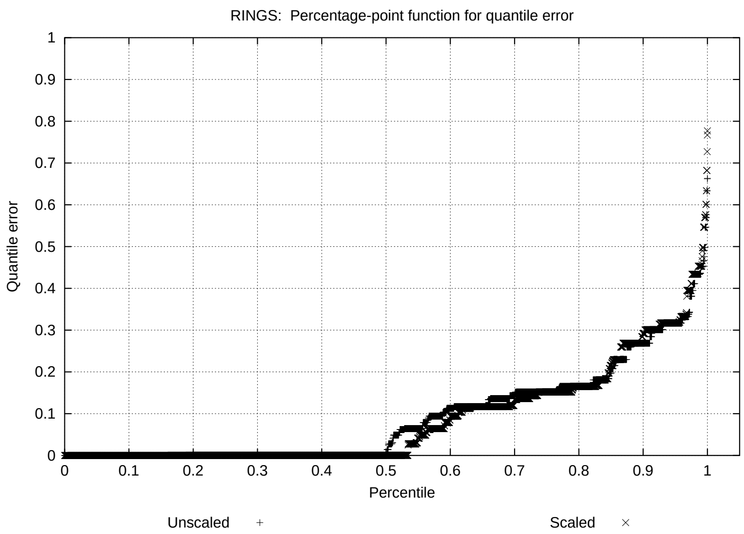


Figure 5.3: Percentage-point curves for quantile error in the unscaled and scaled cases for the `rings` attribute of the `abalone` set. The curves are step functions because `rings` has few distinct values.

Outliers The `abalone` set contains notable outliers. For instance, the model for the scaled version of `height` reports a quantile error of 0.79615 on one tuple; the next worst for that model is 0.63438. The two generating tuples claim to represent the following slightly odd abalones:

- A female abalone with a height of 3mm, a length of 127mm, a diameter of 99mm, and a weight of 231g – very long and wide, but incredibly flat. This is row 1175 in the data set.
- A female abalone with a height of 226mm, a length of 91mm, a diameter of 71mm, and a weight of 66.4g – very large, but among the lightest in the bunch. This is row 2052 of the data set.

These two tuples are both strange enough to be among the top five worst errors for the `shell weight` model in the unscaled case. Figure 5.4 shows just how extreme these two tuples are when considering height and whole weight.

In both the unscaled and scaled cases, the worst result for the `shell weight` belongs to row 3997, one of two infant abalones with heights of 0 – the other being row 1258. That's impressively small for abalones with non-zero length and diameter, weight and ring count.

Attribute	**Bit cost**	$\sum$ **Q error**^{2}	**Nodes**
`Length`	N/A	19.817	5
`Diameter`	N/A	22.237	1
`Height`	N/A	51.434	3
`Whole weight`	N/A	7.514	5
`Shell weight`	N/A	37.633	3
`Rings`	20,016	93.721	3

Table 5.2: Errors for the `abalone` set, with nonlinear scaling. Bit costs reflect the tree cost as well as errors, while quantiles do not. Comparison with the results in the unscaled case show that in both cases, the models are actually quite good in general.

In short, `abalone` is a data set in which for most of the attributes most of the values can be predicted quite well in both the unscaled and scaled cases according to such statistics as the median quantile error; but some tuples deviate extremely so from the mainstream.

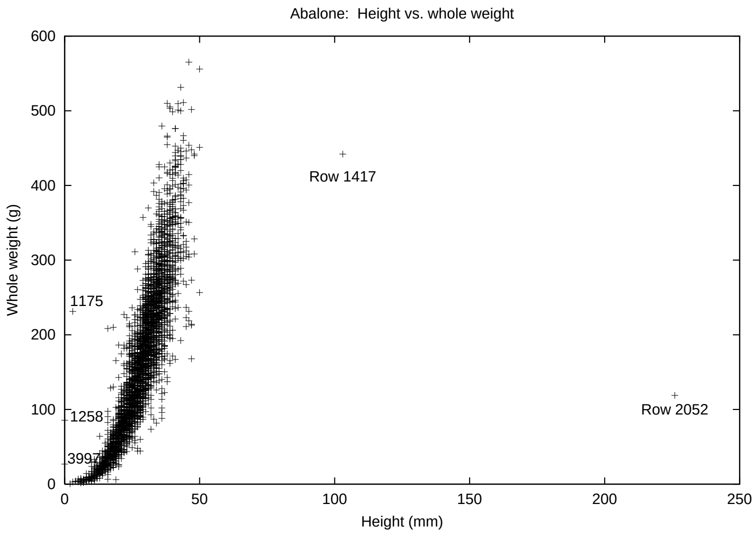


Figure 5.4: Abalone: Height versus whole weight. While there is a bit of a general trend, there are a number of extreme outliers.

5.3.2 Baseball

The 'baseball' data set contains 365 17-dimensional tuples. Each tuple contains a set of statistics for an individual player and the 1996 Major League Baseball season. These statistics listed were those in Table 5.3.

Of these, the `avg` stats are continuous attributes and the rest contain only non-negative integers.

Without nonlinear axis scaling Without scaling, two attributes were retained – `CS` and `E`. Table 5.4 summarizes the quantitative results both the unscaled and scaled models, all of which are single-node trees. Many of the attributes contain only integers, so bit costs based on traditional compression-related objectives do apply; these costs as listed correspond to the bit costs of trees selected with this specific objective in mind. While more complex trees might have enabled better prediction, with the small amount of data the regression tree algorithm is reluctant to split.

BA	Batting avg	3B	Triples
SLG	Slugging pct	BB	Bases on balls
OBA	On-base avg	HR	Home runs
G	Games	BB	Bases on balls
AB	At-bats	SO	Strikeouts
R	Runs	SB	Stolen bases
H	Hits	CS	Caught stealing
TB	Total bases	E	Errors
2B	Doubles		

Table 5.3: Statistics in the baseball data.

Triples: easy to predict Certain of these results deserve explanation. The number of triples, for instance, is most likely relatively easy to predict due to the extreme skew. Fully 133 players had no triples, while 92 had only one triple, and player counts fall sharply as the number of triples increases; none had more than 10. The median quantile error for 3B was rather low at 3.7538×10^{-2}; other than for SB at 7.4380×10^{-2}, every other median quantile error exceeded 0.15 and three – BA, SLG and OBA exceeded 0.22. Recall that a constant median predictor should have a median quantile error of 0.25, or fewer when there are duplicate values.

With nonlinear axis scaling In the case of nonlinear scaling, two attributes were selected – the on-base average, after a square root and scaling derived from the estimation of a two-component mixture of normals; and strikeouts, scaled with a gamma distribution.

For each model common to both the unscaled and scaled cases except 3B and SB, the sum-of-squared quantile errors has improved; those that increase, do so slightly. Median quantile errors now are all below 0.1 except for BA, SLG and G, which are approximately 0.14293, 0.16291, and 0.11019 respectively. The largest reduction in $\sum Q^2$ appears for BB; the unscaled model

$$\mathtt{BB} \leftarrow 17.8370 + 2.6642 \times \mathtt{CS} + 1.4897 \times \mathtt{E}$$

does poorly compared to

$$\hat{\mathtt{BB}} \leftarrow 6.01926 \times 10^{-2} + 0.29904 \times \hat{\mathtt{OBA}} + 0.53628 \times \hat{\mathtt{SO}}$$

	Unscaled		Scaled	
Attribute	**Bit cost**	**$\sum$ Q error^2**	**Bit cost**	**$\sum$ Q error^2**
`BA`	N/A	28.685	N/A	*15.278*
`SLG`	N/A	30.405	N/A	*17.342*
`OBA`	N/A	29.684	`(input)`	
`G`	4,698	17.656	*4,275*	*9.0864*
`AB`	6,089	16.895	*5,623*	*7.7681*
`R`	4,359	17.107	*3,896*	*6.4066*
`H`	4,848	16.430	*4,348*	*7.9507*
`TB`	5,543	20.155	*4,909*	*6.0356*
`2B`	3,319	16.135	*3,009*	*7.7607*
`3B`	*1,784*	*3.8928*	1,934	4.3396
`HR`	*3,333*	23.002	2,940	*5.8189*
`RBI`	4,456	21.428	*3,992*	*6.3872*
`BB`	4,154	21.568	*3,743*	*5.1402*
`SO`	4,584	20.830	`(input)`	
`SB`	*2,507*	*12.449*	3,073	13.810
`CS`	`(input)`		2,306	8.7654
`E`	`(input)`		2,694	12.318

Table 5.4: Errors for the `baseball` set, without and without nonlinear scaling. *Emphasized* text indicates the better performance when the same attribute is modeled in both the unscaled and scaled cases. All trees were single-node; with small sets, the regression tree builder needs to see substantial accuracy improvements to accept a split. Bit costs reflect the tree cost as well as errors but not the cost of the inputs, and is measured on trees specifically chosen by bit cost rather than the quantile metric.

where $\hat{\texttt{BB}}$ has been scaled via a truncated lognormal, $\hat{\texttt{OBA}}$ by a $\frac{1}{2}$-power Box-Cox transformation and a two-component normal, and $\hat{\texttt{SO}}$ by a gamma distribution. The latter formula has a median quantile error of approximately 6.6115×10^{-2} versus 0.1860 in the unscaled case.

Distribution of errors Figure 5.5 shows percentage-point curves for the quantile error in the case of `TB`, for both the unscaled and scaled versions. As with most of the other such graphs for the `baseball` set – with `3B` being a bit of a step function due to the low range – the curve in the scaled case shows a slow and steady increase for most of the errors, and a sharper increase for the worst cases. In the unscaled case, the rate of increase is much more even; thus, the great bulk of the quantile errors are lower in the scaled case despite the fact that both regression trees have but a single node.

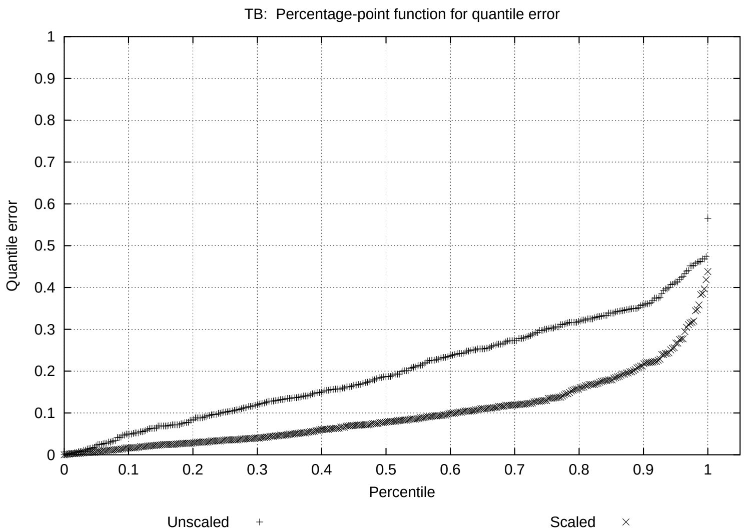


Figure 5.5: Percentage-point curves for quantile error in the unscaled and scaled cases for the `TB` attribute of the `baseball` set. The large gap between the unscaled (upper) and scaled (lower) curves show that while the best- and worst- case errors are similar in magnitude for both, at any other given percentile the scaled case has lower errors. For instance, 90% of the quantile errors for the scaled case are below 0.25; only about 65% of those for the unscaled case are below that same threshold.

Table 5.4 shows that of the 13 attributes modeled in both the unscaled and scaled cases, all the models keep the same number of nodes in both cases and the models with scaling are more accurate in 11 of them. Overall, this data set supports the contention that the added flexibility can improve modeling performance. It may also be noteworthy that for many of the attributes, the bit cost metric and the quantile score agree on whether or not nonlinear axis scaling helped despite making fundamentally different judgments on how to evaluate errors.

5.3.3 Building

The `building` set comes from the `PROBEN1` benchmark collection [Prechelt, 1994], which in turn cites the original Great Energy Predictor Shootout[Kreider and Haberl, 1993]. The version used includes 4,208 tuples featuring nine attributes. `record_id` is ignored; `hour` is an integer; the environmental conditions `temp`, `humid`, `solar` and `wind` are continuous. The three objectives for the Shootout, `wbe` (whole-building electrical energy), `wbcw` (whole-building cold water) and `wbhw` (whole-building hot water) are also continuous attributes.

Without nonlinear axis scaling Without using nonlinear scaling, the attribute selector keeps `temp` and `wind` as inputs.

Attribute	Bit cost	$\sum$ Q error^2	Nodes
`hour`	85,430	2.9319×10^2	1
`humid`	N/A	1.9975×10^2	5
`solar`	N/A	4.9109×10^2	3
`wbe`	N/A	3.5245×10^2	1
`wbcw`	N/A	74.649	1
`wbhw`	N/A	62.211	3

Table 5.5: Errors for the `building` set, without nonlinear scaling. Bit costs reflect the tree cost as well as errors, while quantiles do not.

Table 5.5 summarizes the quantitative results for the models generated without the use of nonlinear scaling. Median quantile errors ranged from 6.9159×10^{-2} for `wbcw` and 8.3675×10^{-2} for `wbhw`, to 0.22201 for `wbe`.

With nonlinear axis scaling In the scaled case, three attributes are selected – `hour`, linearly normalized only; `humid`, with a quarter-power Box-Cox transformation and scaling based on a mixture of normals; and `wind`, no transformation.

Table 5.6 shows the tree sizes and sum-of-squared quantile errors for the resulting models. The comparative results are mixed; the aggregate quantile scores for `wbcw` and `wbhw` have worsened, while the same scores for `wbe` and `solar` have improved. The median quantile errors have become more even; other than `solar`, which had a median quantile error of 6.0054×10^{-2}, the medians range from 0.11529 for `temp` to 0.14946 for `wbhw`.

Attribute	$\sum$ **Q error**^{2}	**Nodes**
`temp`	1.4224×10^2	5
`solar`	46.270	9
`wbe`	2.4256×10^2	5
`wbcw`	1.5121×10^2	1
`wbhw`	1.8242×10^2	1

Table 5.6: Errors for the `building` set, with nonlinear scaling. All of the modeled attributes were continuous; hence no bit costs are listed. Compare with 5.5 regarding `solar`.

Distribution of errors Figure 5.6 compares percentage-point curves for the quantile errors when in the unscaled and scaled cases of modeling `solar`. The scaled error has far lower quantile error at every percentile possibly excepting the smallest errors.

5.3.4 CIA World Factbook (2001)

Extracting `area` (in km^2; continuous) and `population` (in individuals) from the 2001 edition of the CIA World Factbook [Central Intelligence Agency, 2001] yielded a set with 235 tuples and the two specified attributes. Most tuples correspond to nations, although some belong to territories and other entities.

Without nonlinear axis scaling In the unscaled case, the system chooses to retain `population` as the input. The model for `area` uses only one node and results in a sum-of-squared quantile error of 16.901; individual quantile errors range from 4.8381×10^{-3} to 0.52949, with the median at 0.20891. In other words, the unscaled `area` predictor is not much better than a ideal constant-median predictor.

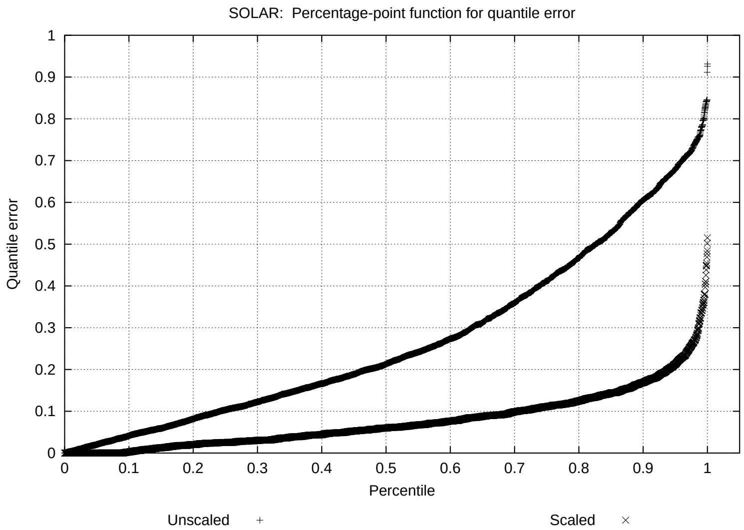


Figure 5.6: Percentage-point graphs for quantile error in the unscaled and scaled cases for the `solar` attribute of the `building` set. The curve for the scaled case is *much* lower than the curve for the unscaled case. Over 90% of the quantile errors from the scaled case are below the *median* quantile error of the unscaled case.

With nonlinear axis scaling With nonlinear scaling enabled, the system again selects `population` as an input, but with a $\frac{1}{3}$-power Box-Cox transformation and a gamma-based transformation. The single-node tree for modeling `area` achieves a sum-of-squared quantile error of 6.0886; individual quantile errors range from 1.5408×10^{-4} to 0.71214, with the median at 9.1444×10^{-2}. The expanded range is interesting in that not only have most of the errors declined, but the magnitude of the highest observed quantile error has increased dramatically – and usefully.

It may be noteworthy that while in the unscaled case, the progression of errors when sorted by magnitude appears smooth, while the scaled case the magnitude of the worst quantile error, 0.71214, greatly exceeds that of the second-worst, 0.45605. Examination of the data suggests that this worst case belongs to the territory of Greenland; with its area of 2,175,600 km^2 and a population of 56,352 this extremely sparsely-populated territory *should* be an outlier. In the unscaled case, it isn't very much of one, with a quantile error of 0.41404 leaving it below 34 others.

Outliers The low dimensionality of this set makes it feasible to graph the effects of scaling. Figures 5.7 and 5.8 show the before-and-after comparison; in the scaled case, the axes graphed are the scaled version selected by the attribute partitioning scheme and the scaled version associated with the best-performing model.

Figures 5.9 and 5.10 show some annotations for the unscaled and scaled cases, respectively. In the former case, the only points that stand out are those with unusually high area; in the latter case, we can actually label the ten-worst modeled points according to the quantile metric. Looking at the latter it is clear that Greenland is an extreme outlier; indeed, it receives the single-highest quantile error after applying a $\frac{1}{2}$-power Box-Cox and gamma distribution to the area, and a $\frac{1}{3}$-power Box-Cox and gamma to the population.

Mongolia, another sparsely populated state with an area of 1.565×10^6km^2 and a population of $2,654,999$, accounts for the second-worst score. Namibia, Mauritania, and Svalbard round out the top five; all seem to have unusually low population densities. By comparison, the People's Republic of China is modeled reasonably well, with a quantile error of only 4.5272×10^{-2}; it is highly unusual in terms of both area and population, but the relationship between the two is normal. The same holds for India; the quantile error is a mere 3.4981×10^{-2}. In the unscaled case, China and India are modeled with quantile errors of 3.1532×10^{-2} and 3.7028×10^{-2}; acceptable in both cases and a push when comparing.

On the other hand, in the unscaled case the worst-modeled area is that of Monaco, with a quantile error of 0.52949, followed closely by the Vatican at 0.52498, Gibraltar at 0.52443, Macau at 0.52267, and Tokelau at 0.51221. In the scaled case, these tiny areas received quantile errors of 0.20639, 4.98682×10^{-2}, 0.19175, 0.27043, and 0.04209, respectively. These tend to be tiny states; however, Tokelau's population density of 144.5 people per km^2 is not extreme, with 68 densities greater, 165 less, and one other approximately identical – Andorra. In neither the unscaled case nor the scaled case is Andorra considered an outlier.

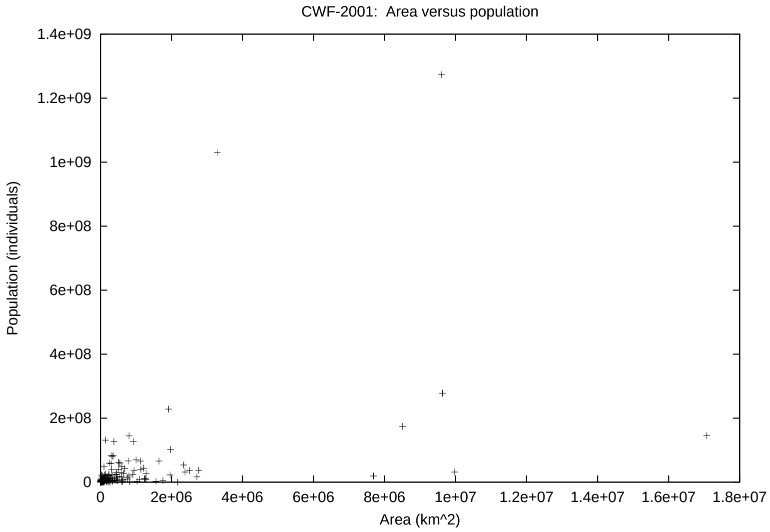


Figure 5.7: Area and population data from the 2001 edition of the CIA World Factbook. It would not be advisable to estimate the relationship between `area` and `population` from this scaling.

Distribution of errors Figure 5.11 shows the percentage-point functions for the quantile error when modeling `area`, in both the unscaled and scaled cases. The unscaled errors are again usually much lower across the board, although the difference narrows with the very extremes; in fact, the worst case is 'worse' in the scaled model. This might be explained by the exceptional nature of Greenland when handled as a separate entry from Denmark, and its incredibly low population for its huge area.

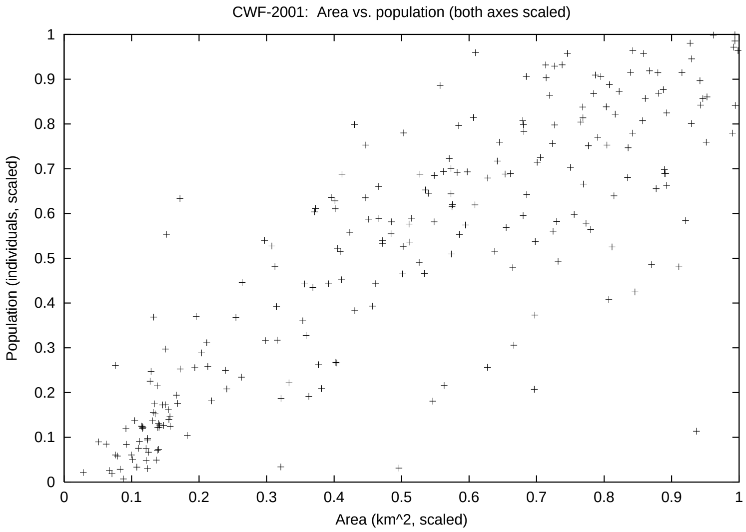


Figure 5.8: Area and population data from the 2001 edition of the CIA World Factbook, after nonlinear axis scaling. While the trend is far from perfect, this graph does suggest a broad and positive correlation between `area` and `population`; and also suggests the existence of a few extreme exceptions.

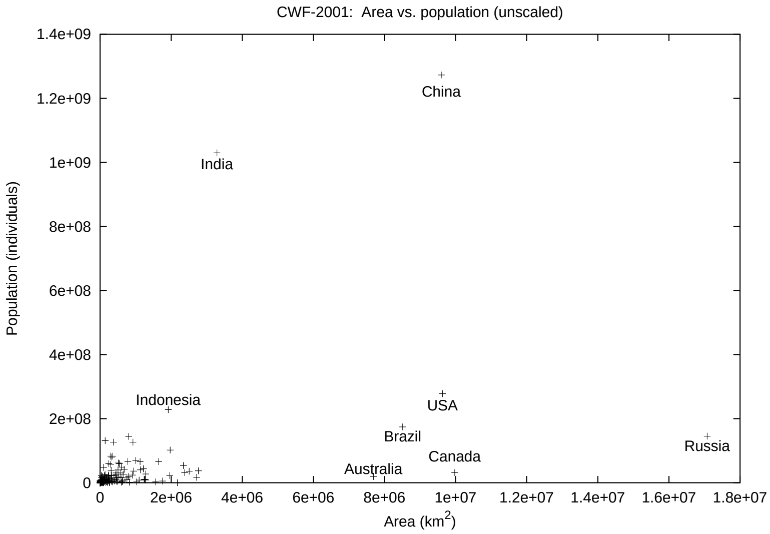


Figure 5.9: Before axis scaling, with annotations. One might classify all of the named regions as outliers, using a traditional cluster-based definition.

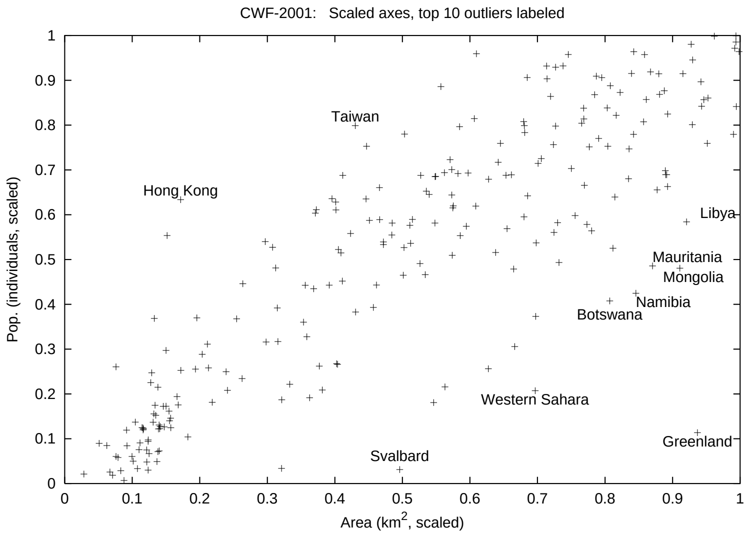


Figure 5.10: After axis scaling, with annotations. The named points do not necessarily correspond to extrema, but to nations which deviate from the general trend relating `area` and `population`. Hong Kong, for instance, is extremely densely populated.

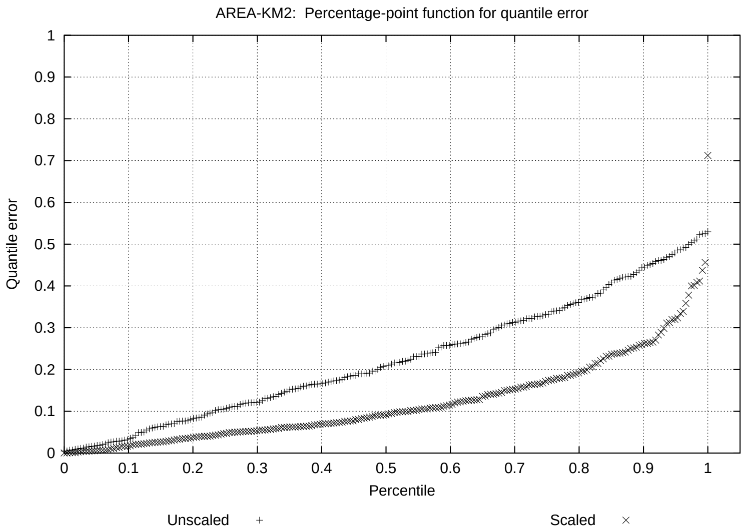


Figure 5.11: Percentage-point curves for quantile error in the unscaled and scaled cases for the `area` attribute of the `Factbook` set. The much lower curve for the scaled quantile errors shows the degree of improvement; approximately 80% of the quantile errors in the scaled case fall below the `median` quantile error of the unscaled case.

5.3.5 DJ30

The `DJ30` set is the `DJ1985-2003` submitted by Danilo Careggio to `StatLib` [Careggio]. This set contains stock data for each stock that was a component of the Dow Jones 30 Industrials in October 2003, split-adjusted, over a period consisting of 2,529 trading days from January 2, 1985 to October 30, 2003. While one attribute was labeled `CO`, I refer to it as `KO` on the suspicion that – `CO` not having been a DJIA component while `KO` was – this was a labeling error. The truth of the matter has few ramifications other than the fact that `CO` and `KO` operated in different markets and presumably have different dependencies. The data set description indicates that the data was originally collected from `Yahoo!`.

From this set, I used the split-adjusted closing prices. While the sequential nature of the data would have allowed it, no experimentation was performed with exploiting temporal correlations. For instance, a model for the closing price of American Express (`AXP`) did not use any historical prices for American Express, and could use only the closing prices of the other Dow Jones 30 components for that same day. One could obviously perform considerable experimentation attempting to predict thirty concurrent and non-independent time series.

Without nonlinear axis scaling Without nonlinear scaling, only a single stock was chosen as an input – IP, or International Paper. Table 5.7 lists the tree sizes and sum-of-squared quantile errors for the remaining attributes.

Let us consider the results from various angles. First, most of the sum-of-squared quantile errors are on the order of 100 to 200, with a few notable exceptions. The least occurred with DuPont (`DD`); there, a 5-node tree had a maximum quantile error of 0.48280, and a median quantile error of 7.3211×10^{-2}. The model for Caterpillar (`CAT`) is almost as accurate, with median and maximum quantile errors of 9.2970×10^{-2} and 0.49715 respectively. For Proctor & Gamble (`PG`), the slightly worse score reflects median and maximum quantile errors of 9.99612×10^{-2} and 0.57122; Philip Morris (`MO`; now Altria), 9.7920×10^{-2} and 0.54149. By the time the score reaches that of Johnson & Johnson (`JNJ`), the median quantile error has reached 0.14822, more than twice that of DuPont.

With nonlinear axis scaling With nonlinear scaling enabled, the system selects two attributes – International Paper, with a third-root Box-Cox and a truncated normal; and JP Morgan, with a mixture of normals.

Stock	$\sum \mathbf{Q}^2$	**Nodes**	**Stock**	$\sum \mathbf{Q}^2$	**Nodes**
AA	1.0201×10^2	11	JNJ	1.0607×10^2	9
AXP	1.1884×10^2	7	JPM	97.925	7
BS	1.3244×10^2	3	KO	1.4506×10^2	5
CAT	52.337	7	MCD	1.0858×10^2	5
CI	1.1298×10^2	7	MMM	1.0348×10^2	9
DD	43.6504	5	MO	75.273	9
DIS	1.2681×10^2	3	MRK	1.1442×10^2	11
EK	1.7651×10^2	5	MSFT	1.1252×10^2	3
GE	1.2157×10^2	9	PG	64.729	5
GM	1.2142×10^2	1	SBC	1.1358×10^2	1
HD	1.2567×10^2	7	T	1.7226×10^2	5
HON	76.534	7	UTX	1.0692×10^2	9
HPQ	1.7325×10^2	11	WMT	1.5867×10^2	5
IBM	1.0122×10^2	5	XOM	1.2756×10^2	11
INTC	1.2795×10^2	11			

Table 5.7: Errors for the DJ30 set, without nonlinear scaling. All of the modeled attributes were continuous, hence no bit costs are listed. BS is Bethlehem Steel, acquired after the period of data collection. The models range considerably in size and accuracy.

Table 5.8 presents the model statistics after enabling nonlinear scaling. In all 28 models common to both the scaled and unscaled cases, the sum-of-squared quantile error has dropped. The median of the median quantile errors for all the models is down from 0.14890 to 7.4698×10^{-2}; the maximum median quantile error dropped from 0.20833 to 0.14980, with these maximum medians belonging to `WMT` in both the scaled and unscaled cases.

Stock	$\sum \mathbf{Q}^2$	**Nodes**	**Stock**	$\sum \mathbf{Q}^2$	**Nodes**
`AA`	57.609	11	`INTC`	29.184	3
`AXP`	42.130	9	`JNJ`	68.910	13
`BS`	70.430	13	`KO`	92.650	13
`CAT`	46.381	3	`MCD`	28.461	9
`CI`	41.522	11	`MMM`	75.824	9
`DD`	30.926	1	`MO`	49.874	11
`DIS`	23.447	11	`MRK`	34.258	11
`EK`	1.3611×10^2	9	`MSFT`	29.300	11
`GE`	42.347	5	`PG`	55.928	5
`GM`	27.394	5	`SBC`	21.837	11
`HD`	45.104	3	`T`	1.0389×10^2	9
`HON`	19.474	7	`UTX`	57.333	11
`HPQ`	35.822	9	`WMT`	91.253	5
`IBM`	67.7805	5	`XOM`	51.123	13

Table 5.8: Errors for the `DJ30` set, with nonlinear scaling. All of the modeled attributes were continuous, hence no bit costs are listed. Compare with Table 5.7; the sum-of-squared quantile errors are lower in *every* case, with some increase in model complexity.

Studying the time series The sequential nature of the data, the dates, and the availability of news articles about these stocks and the stock market in general give us an opportunity to study these data in more detail, and in some cases to attempt to explain any observations. Consider Figure 5.12. The two curves in that figure show the daily median quantile error across all output attributes in the unscaled and scaled cases as time goes by. While the scaled case does not completely lie below the unscaled case, it is generally lower and never reaches the heights that the scaled case. Since these are median quantile errors across a group of outputs, a sudden spike would likely correspond to either a significant event affecting an input attribute, or affecting at least half the outputs. For instance, the largest spikes in the unscaled case occur around September to October 2000; there does not seem to be a similar spike in for the scaled case. This may be reflecting news specific to `IP`, which at the time was dealing with the aftermath of acquisitions and capacity reduction; while it was an input in the scaled space as well, the presence of another input `JPM` may have reduced this. Alternately, the scaling of `IP` may have mitigated its effects.

Temporal locality In the scaled case, the errors of quite a few models increase sharply around the middle of September to early October of 2002. This may be related to problems afflicting `JPM` at that time, notably fallout from its well-publicized involvement and possible liability in the Enron, Adelphia, and MCI Worldcom scandals. Many of the worst quantile errors – for instance, the top 9 errors for `IBM`; the top 3 for `MO`; 15 of the top 20 for `PG` – fell between September 17 and October 9.

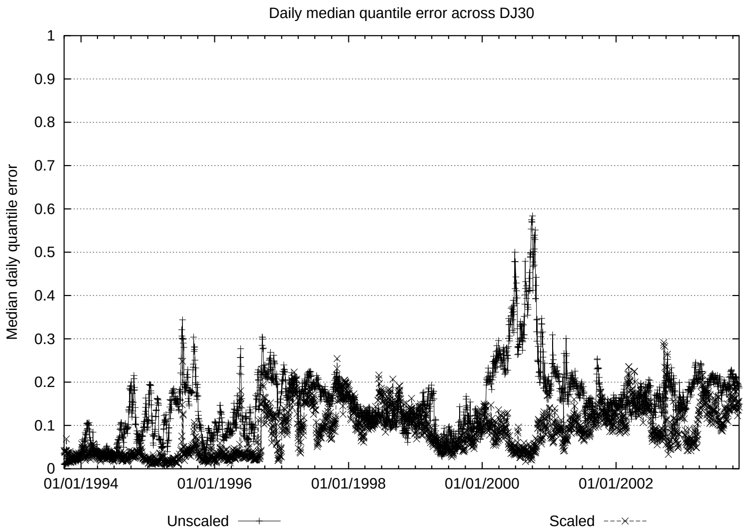


Figure 5.12: Daily median quantile error across all the `DJ30` outputs, in both the unscaled and scaled cases. The curve for the `scaled` case is more stable and usually lower, while the unscaled curve has occasional sharp spikes indicating substantial problems with at least half the models.

One can also examine the errors of individual models. For example, let us consider the 5-node regression tree that models `DD`. 18 of its top 25 worst errors occur from August 8, 2001 to September 6, 2001. These errors, in the range $(0.3201, 0.3872)$, are all much greater than the median quantile error of 6.0691×10^{-2}. In June, the company had announced the sale of its pharmaceutical unit to Bristol-Myers Squibb; in late July, they announced that job cuts would be larger than expected, and admitted to losing money in a quarterly report. For another model, `CAT`, 8 of the 13 worst errors occur on the eight trading days from April 9, 2003 to April 22, 2003. All of these errors exceed 0.3700 in quantile space; the median was 7.3458×10^{-2}. It may be noted that in early April S&P downgraded `CAT` to 2 stars (avoid) from 3 stars (hold), labeling its rise in share price an overreaction [BWO, 2003]. The two worst quantile errors for `SBC`, of magnitudes 0.5077 and 0.5362, fall on the 20th and 21st of September, 2001, and vastly exceed the third worst error of magnitude 0.3551. On the 21st, it had announced plans to purchase outstanding shares of Prodigy Communications; this may have relevance.

Honeywell In the scaled case, the most accurate model according to the sum-of-squared quantile error belongs to Honeywell (`HON`); in the unscaled case, the corresponding model scores decently relative to the others but is not the best. Figures 5.13 and 5.14 each show two curves and a set of impulses where the two curves correspond to the actual and estimated split-adjusted closing prices, as per the left Y axis; the impulses track quantile errors for the daily estimations on the right Y axis. In the scaled case, the estimates appear to track the actual closing prices much better than in the unscaled case; this improved performance makes it much more interesting to examine the exceptions. Figure 5.15 shows the actual model in scaled space.

According to the quantile metric, the scaled model's worst mispredictions occurs on October 23, 2000 with a score of 0.42820; this is the peak of a series of quantile errors exceeding 0.1 starting on October 9, 2000 and ending by December 21, 2000 with respites on October 13 and 18. On October 23, it was announced that General Electric had agreed to buy Honeywell International, beating a prior bid by United Technologies [Hackley]. Rumors of the `UTX` bid had been officially confirmed by October 19, 2000 [Johnson, 2000]. Another period of interest might be April 10 to April 17 of 2002, when the quantile errors for the scaled model briefly increased, peaking at 0.24880 on April 12; Honeywell did gain sharply on April 10, enough to be mentioned in articles [Greenwald, 2002], but searching did not reveal any specific reason.

DuPont After nonlinear scaling, the "best bargain" model would have to be `DD`, or DuPont. The model for `DD` has an acceptably low sum-of-squared quantile error

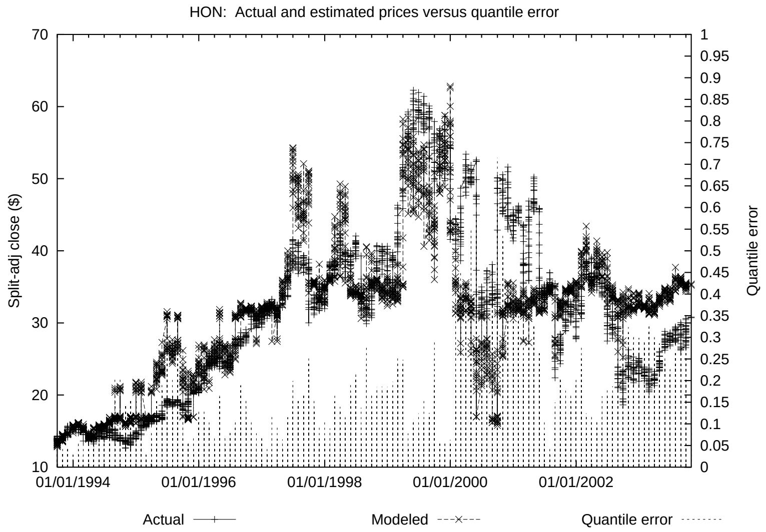


Figure 5.13: Actual and modeled prices for Honeywell `HON` using the left Y-axis; daily quantile error, using the right Y-axis. The curves diverge considerably at times, thus accounting for the high quantile errors noted by the taller impulses.

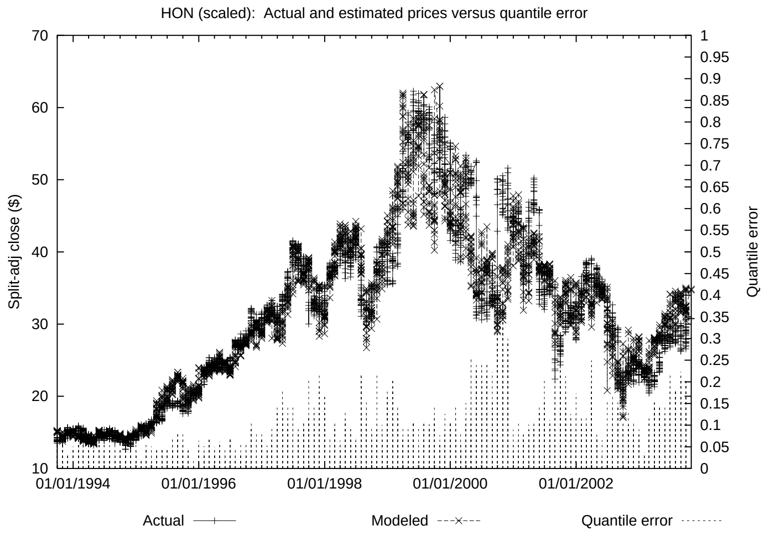


Figure 5.14: Actual and modeled prices for Honeywell `HON` (scaled) using the left Y-axis; daily quantile error, using the right Y-axis. The curves track each other very well most of the time, indicating good prediction.

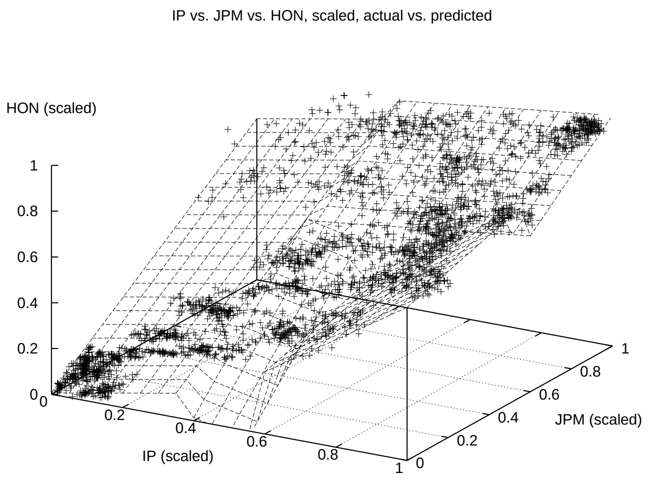


Figure 5.15: Scaled model for `HON` using `IP` and `JPM`. The observations, indicated by the crosses, are well-modeled by the surfaces depicting the regression tree.

of 30.926, a median quantile error of 6.0691×10^{-2}, and a maximum quantile error of 0.3871, making it more accurate than every model in the unscaled case – and with a single node.

With `IP` scaled via a $\frac{1}{4}$-power Box-Cox transformation and a truncated normal, `JPM` scaled via a seven-component mixture of normals, and `DD` scaled via a logarithmic transformation and a mixture of two normals, the "tree" consisted solely of one equation:

$$\hat{\mathtt{DD}} \leftarrow -1.2420 \times 10^{-2} + 0.66000 \times \hat{\mathtt{IP}} + 0.35426 \times \hat{\mathtt{JPM}}$$

It's not the most accurate model in the list – the scaled models for `DIS`, `GM`, `HON`, `INTC`, `MCD`, `MSFT`, and `SBC` have lower $\sum Q^2$ error – but it's still quite accurate and is the least complex. A possible explanation lies in the fact that DuPont and International Paper both count paper goods as nontrivial parts of their product lines.

Eastman Kodak The corresponding graphs for the scaling's worst case, Eastman Kodak (`EK`) are Figures 5.16 and 5.17. In the unscaled case, the model has essentially collapsed to the point of in places being practically a constant predictor. The unsurprising result is that the quantile errors are very high. The scaled case fares better, but still completely fails with certain trends – such as modeling a steep decline instead of a sharp rise circa July 1998 with the greatest discrepancy perhaps on October 8, 1998 when the model predicts a closing price of \$31.25 instead of the actual price of \$64.55. It seems relevant that a very sharp climb in share price had started on July 15, 1998 when `EK` closed at \$67.23 a share (split-adjusted) – much higher than the \$60.10 a share of the preceding day. On that day, `EK` reported a very positive earnings surprise – \$1.38 a share for the second quarter, versus expectations of \$1.13 and actual earnings of \$1.11 for the same quarter in the previous year. The stock peaked a week later at \$71.71 a share, and by March 19, 1999 had fallen to \$55.35 a share – close to the scaled model's estimate of \$53.82 s share based on `IP` and `JPM`.

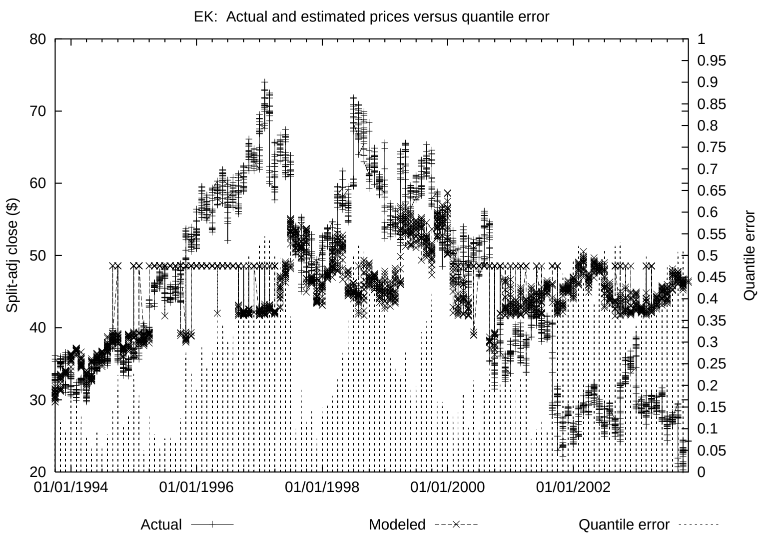


Figure 5.16: Actual and modeled prices for Eastman Kodak `EK` using the left Y-axis; daily quantile error, using the right Y-axis. The curves diverge rather often and significantly, and the high impulses indicate substantial quantile errors.

SBC Communications SBC Communications (`SBC`) offers an interesting example of the improvements made possible by nonlinear scaling. Tables 5.7 and 5.8 note that the sum-of-squared quantile error has improved significantly for `SBC`; the median quantile error has dropped from 0.16600 to 5.2031×10^{-2}. On the other hand, the

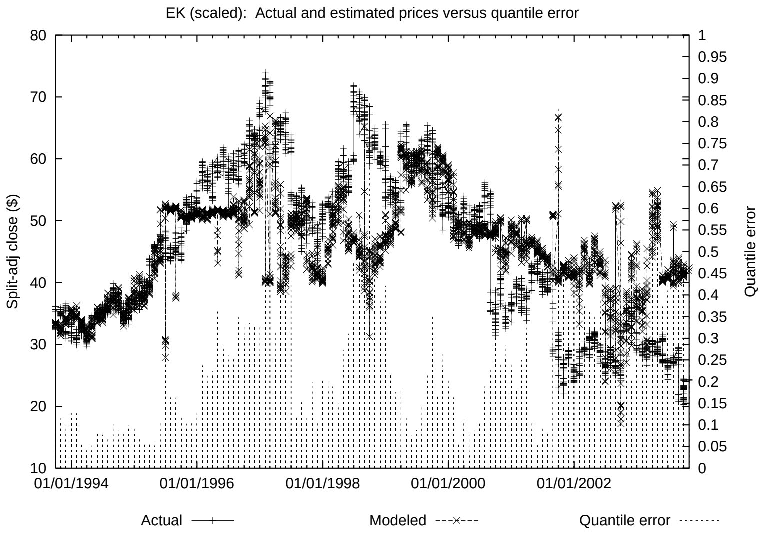


Figure 5.17: Actual and modeled prices for Eastman Kodak `EK` (scaled) using the left Y-axis; daily quantile error, using the right Y-axis. While there are still periods of significant divergence and high quantile errors, the scaled model tracks somewhat better than the unscaled model shown in Figure 5.16.

scaled model shown in Figure 5.18 requires two inputs and eleven nodes instead of one input and one node.

Figure 5.19 shows the closing prices of `IP` and `SBC`, unscaled. A cursory examination shows that `SBC`'s closing price must depend on something other than `IP`; since the unscaled case resulted in selecting only `IP` as an input, `SBC` could *not* have been modeled well. It therefore becomes natural to ask whether or not the inclusion of `JPM` would suffice. Figure 5.20 shows the outcome, should one force this selection and build a regression tree with these inputs and output; the resulting 5-node tree has a median quantile error of 7.1377×10^{-2} and a sum-of-squared quantile error of 30.619; smaller but less accurate than the model derived with nonlinear scaling.

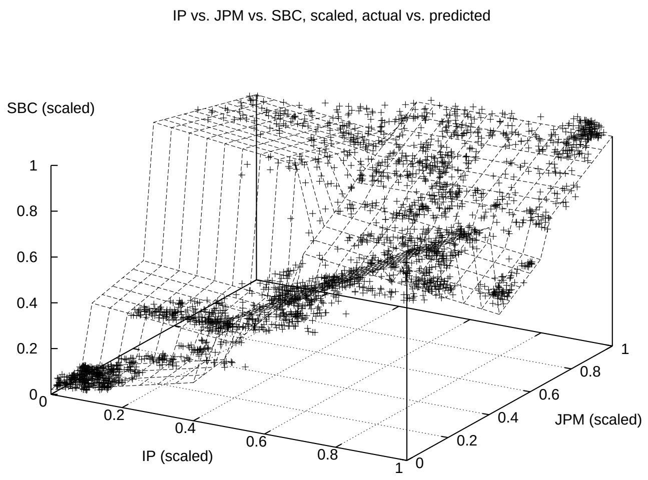


Figure 5.18: `IP` and `JPM` versus `SBC`, scaled. The observations, marked by crosses, are well-tracked by the model.

`INTC` Intel (`INTC`) shows a similar case. As Figure 5.21 shows, the single input chosen in the unscaled case *cannot* be expected to produce a reasonable piecewise linear model; the sum-of-squared quantile error of 1.2795×10^{-2} and the median quantile error of 16.269 reflect this even after using 11 nodes in the regression tree. Compare that with the results of the scaled case using both `IP` and `JPM` as inputs, as per Figure 5.22; with three nodes, the model has a sum-of-squared quantile error of 29.184 and a median quantile error of 6.0731×10^{-2}. For comparison, Figure 5.23

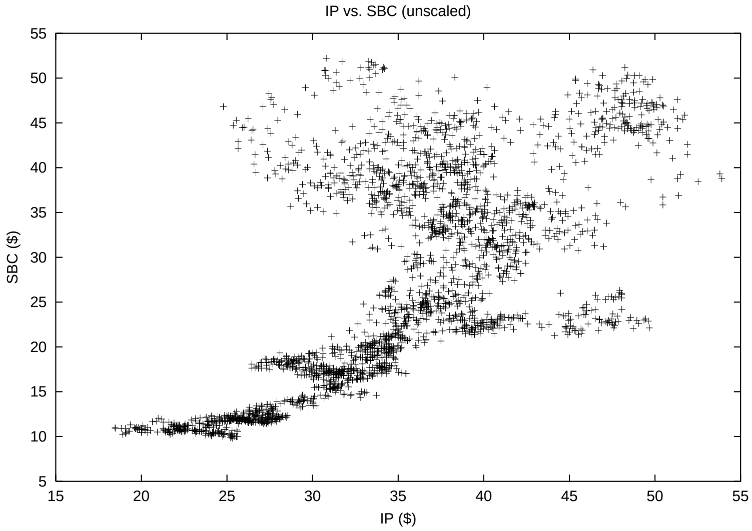


Figure 5.19: `IP` versus `SBC`, unscaled. Clearly one needs more than `IP` to model `SBC`.

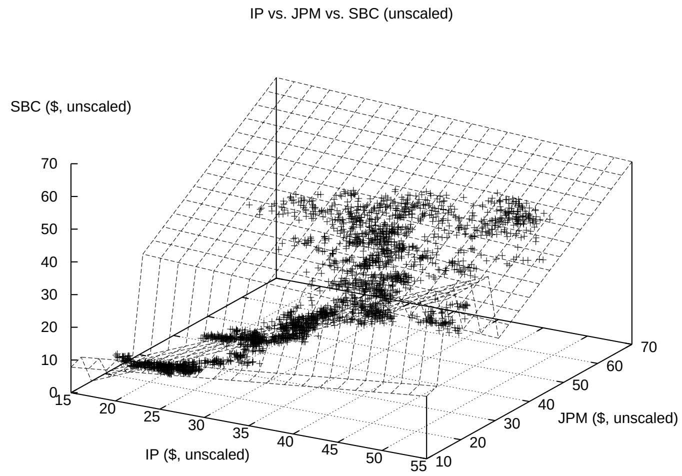


Figure 5.20: `IP` and `JPM` versus `SBC`, unscaled. This model is fairly accurate, but not as accurate as that of the scaled case shown by Figure 5.18.

shows the model that would have resulted if the unscaled case had permitted both IP and JPM as inputs instead of only the former. This five-node two-input model has slightly more complexity than in the scaled case, and a slightly better median quantile error of 5.7527×10^{-2}, but a worse sum-of-squared quantile error of 39.540.

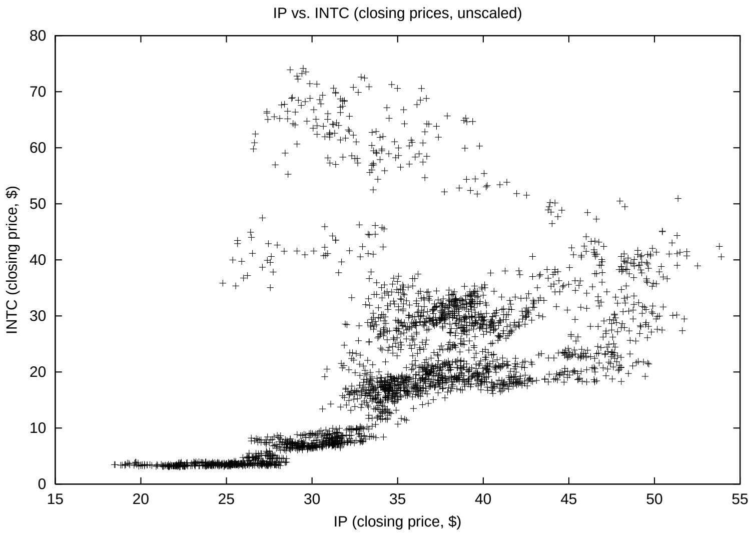


Figure 5.21: IP versus INTC, unscaled. One needs more than IP to model INTC.

Distribution of errors With the number of attributes involved and the nature of the data – some companies and their stock prices are much more related than others, plus some may have had substantial rumors or events specific to them – it seems useful to present multiple graphs comparing percentage-point curves. Figures 5.24, 5.25, 5.26, and 5.27 show them for GM, DIS, CAT and EK respectively. GM presents the usual example of the scaled case dominating the unscaled in terms of lower quantile error throughout. With DIS, the gap seems unusually large; with CAT, the difference is minimal. EK is unusual in that neither the scaled model or the unscaled is all that accurate.

5.3.6 Housing

The housing data set can be found at the UCI Repository [Murphy and Aha, 1994], but originated with the research described in [Harrison and Rubinfeld, 1978]. Sub-

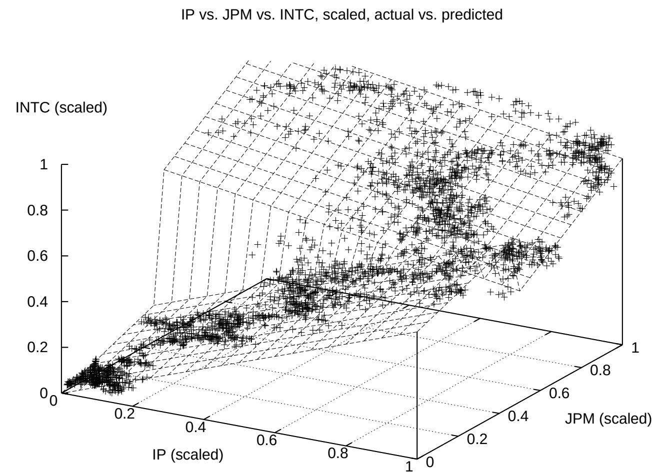


Figure 5.22: `IP` and `JPM` versus `INTC`, scaled. The model's surfaces track the observations quite well.

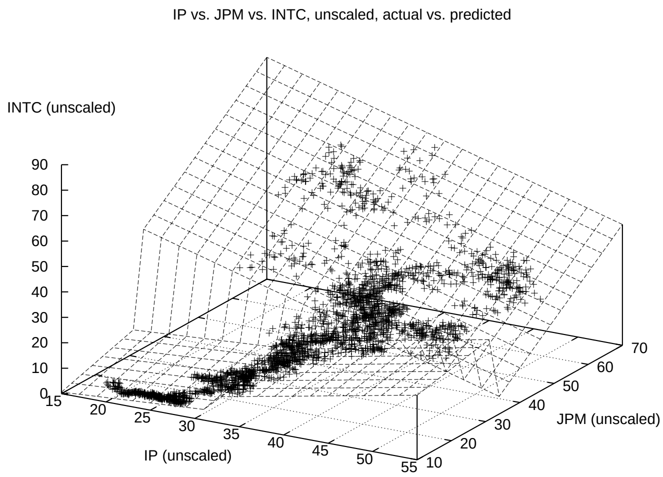


Figure 5.23: `IP` and `JPM` versus `INTC`, unscaled. The model here is more complex than that of the scaled case per Figure 5.22, but with mixed results regarding accuracy – the median quantile error has declined slightly but the sum-of-squared quantile error has increased.

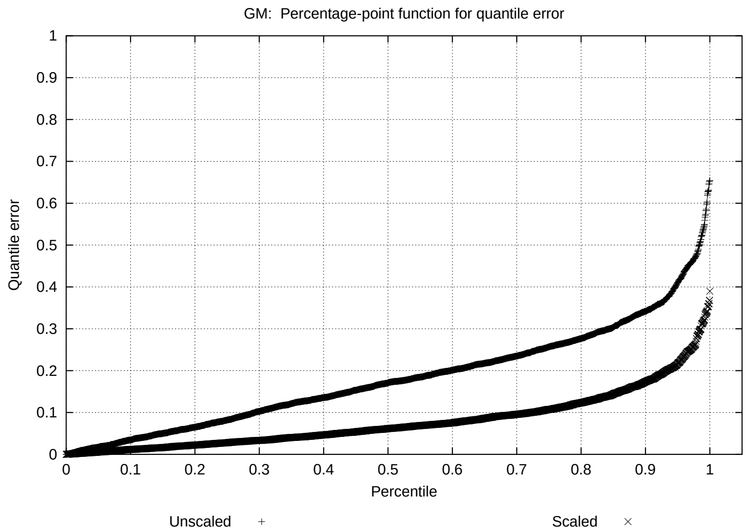


Figure 5.24: Percentage-point curves for quantile error in the unscaled and scaled cases for `GM` from `DJ30`. The scaled case, reflected by the lower curve, produces much lower errors overall than the unscaled case.

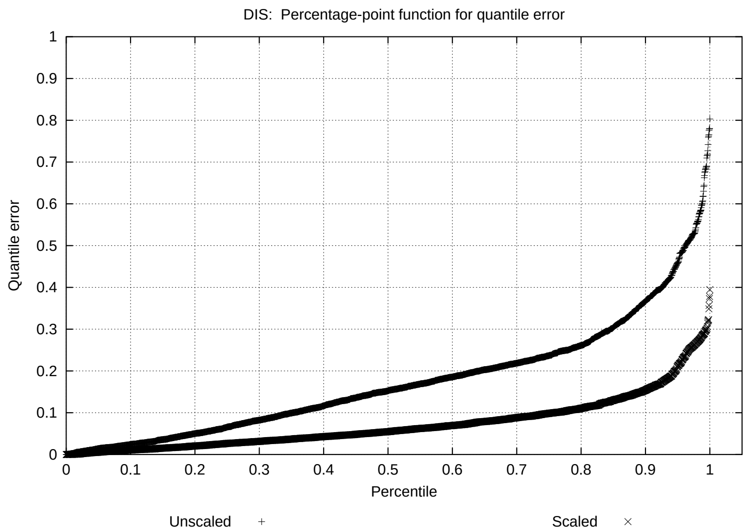


Figure 5.25: Percentage-point curves for quantile error in the unscaled and scaled cases for `DIS` from `DJ30`. Again, the lower curve for the scaled case shows much lower quantile errors than for the unscaled case.

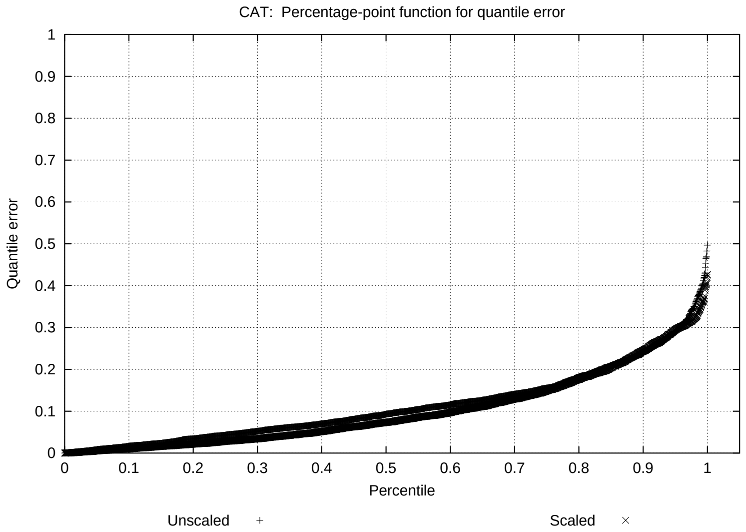


Figure 5.26: Percentage-point curves for quantile error in the unscaled and scaled cases for `CAT` from `DJ30`. The nearly coincident curves occur because the models are approximately equally good.

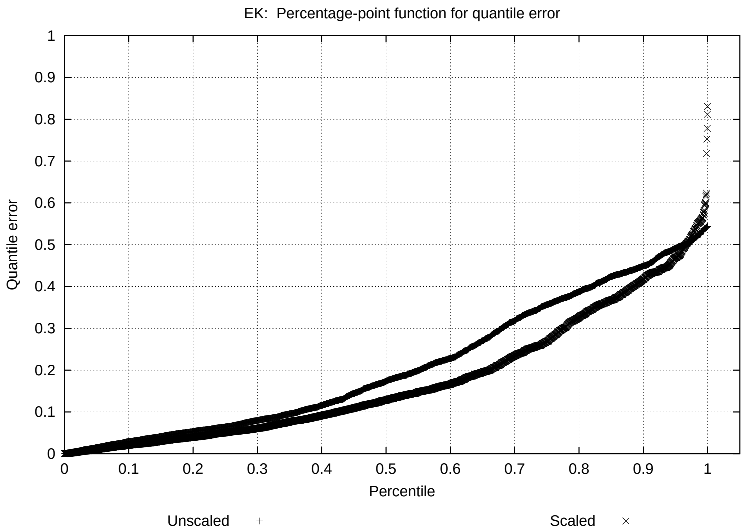


Figure 5.27: Percentage-point curves for quantile error in the unscaled and scaled cases for `EK` from `DJ30`. The curves are similar, and in both cases there is less of a pronounced rise towards the end. Neither model is especially good.

sequent research in the originating domain has questioned the encoding [Gilley and Pace, 1996]. However, since the the point of this testing is to assess the effects of nonlinear scaling on the modeling process, and not to draw conclusions about the Boston housing market circa 1970, let us ignore this and any other disputes regarding its accuracy.

This set contains only 506 tuples, each of which has fourteen attributes. The original problem consisted of predicting median home value based on a variety of factors. Twelve attributes are continuous, including the designated output `MEDV`; one attribute is binary; and one is an integer.

Without nonlinear axis scaling Absent the enabling of nonlinear scaling, the system selected three attributes – `CHAS`, a binary, and thus automatically selected, attribute specifying whether the land borders the Charles River; and `NOX`, a continuous attribute specifying the concentration of nitric oxides in parts per 10 million; and `RM`, the average number of rooms per dwelling.

Table 5.9 gives the basic results for the models in the unscaled case. All the trees are quite small, as expected due to the small size of the data set and the regression tree builder's bias against large trees.

We first examine the few most accurate models. The lowest sum-of-squared quantile error occurs with `ZN`. This apparent best case has a median quantile error of only 1.8746×10^{-3}, and 500 of the quantile errors are 0.23025 or below; the remaining six range from 0.84601 to 1.1675. With `INDUS`, a tree with a single expression, $\text{INDUS} = -0.95378 + 40.922 \times \text{NOX}$, the median error is higher, at 7.9642×10^{-2}, and the maximum is 0.48230 with no outstanding outliers. The table suggests that `LSTAT` should be slightly less accurate with $\text{LSTAT} = 37.461 + 24.453 \times \text{NOX} - 6.0014 \times \text{RM}$; the median quantile error was 0.12140, and the maximum is 0.86884.

Extremely large errors The most interesting cases here are `B` and `AGE`, with their large sum-of-squared quantile errors and high median errors of 0.31245 and 0.16353, respectively. A median quantile error in excess of 0.25 is high indeed; a model that always predicts the true median should score no worse than that threshold. For `B`, 59 of the tuples produce quantile errors in the range of $[1.0607, 9.4382]$. `AGE` also has 24 tuples producing quantile errors in the range of $[1.2612, 3.8620]$. These outliers account for the unusually poor $\sum Q^2$ scores of their models.

Attribute	Bit cost	$\sum$ Q error^2	Nodes
CRIM	N/A	35.724	5
ZN	N/A	7.3285	3
INDUS	N/A	15.511	1
AGE	N/A	2.4482×10^2	1
DIS	N/A	21.470	1
RAD	3991	24.332	1
TAX	N/A	26.492	1
PTRATIO	N/A	29.179	1
B	N/A	1.3684×10^2	1
LSTAT	N/A	17.603	1
MEDV	N/A	21.114	1

Table 5.9: Errors for the `housing` set, without nonlinear scaling. As with the `baseball` data, the small size of the set makes it difficult to justify splitting nodes in the regression tree. Note `B` and `AGE`.

With nonlinear axis scaling The system retains more input attributes after the enabling of nonlinear scaling. Specifically, it chose `CRIM`, with a mixture of lognormals; `CHAS`, identity, as is mandatory for a binary attribute; `RM`, with a fifth-root Box-Cox transformation and mixture of normals; `AGE`, with a mixture of lognormals; and `B`, with a fifth-root Box-Cox transformation and mixture of normals.

Table 5.10 summarizes the quantitative measurements of the models built with the assistance of nonlinear scaling. Of obvious note is that none of the sum-of-squared quantile costs suggests the existence of extreme outliers, unlike `B` and `AGE` in the unscaled case. It's also notable that for *every* attribute modeled in both the scaled and unscaled case, these sum-of-squared quantile errors are lower for the scaled case – sometimes significantly – and the trees are no larger except for `TAX`, which grew from 1 to 3 nodes.

Consider the apparent best and worst cases. With `ZN`, modeled after a quarter-power Box-Cox transformation and estimation using mixture of normals, a single-node tree yielded a tiny median quantile error of 1.4519×10^{-3}, and a maximum quantile error of 0.26535. 382 tuples were modeled with quantile errors below 0.5% The worst case of `PTRATIO` still has a median quantile error of 0.11952 and a maximum quantile error of 0.66749; in the unscaled case, these scores were 0.14510 and 0.79040, respectively. The score for `RAD` has improved significantly; this is corroborated by a median quantile error of 1.9802×10^{-3} (down from 8.1312×10^{-2} in the unscaled case) and a maximum quantile error of 0.31485 (down from 0.70144).

For the curious, the model for ZN simply encodes

$$\hat{\mathtt{ZN}} \leftarrow 0.68972 - 0.24887 \times \hat{\mathtt{CRIM}} + 0.14531 \times \hat{\mathtt{RM}} - 3.5944 \times 10^{-2} \times \hat{\mathtt{B}}$$

where the output ZN has been scaled via a $\frac{1}{4}$-power Box-Cox transformation and a six-component mixture of normals.

Attribute	Bit cost	$\sum$ Q error^2	Nodes
ZN	N/A	2.6516	1
INDUS	N/A	13.282	1
NOX	N/A	8.8122	1
DIS	N/A	12.494	1
RAD	3,049	3.6755	1
TAX	N/A	14.443	3
PTRATIO	N/A	26.575	1
LSTAT	N/A	13.038	1
MEDV	N/A	16.778	1

Table 5.10: Errors for the housing set, with nonlinear scaling. Attributes modeled in both cases have lower sum-of-squared quantile errors, without a corresponding gain in model complexity except for an additional two nodes for TAX.

Distribution of errors The one unusual unusual quantile-error percentage-point function is for RAD, the "index of accessibility to radial highways". For this attribute, the only values in the data set are the integers 1 through 8, and 24. Hence, Figure 5.28 shows a step function.

5.3.7 Liver

The liver disorders set may also be found at the UCI Repository [Murphy and Aha, 1994], which credits it to BUPA Medical Research Ltd. This data set has 345 7-dimensional tuples. Five attributes – mcv, the mean corpuscular volume; alkphos,

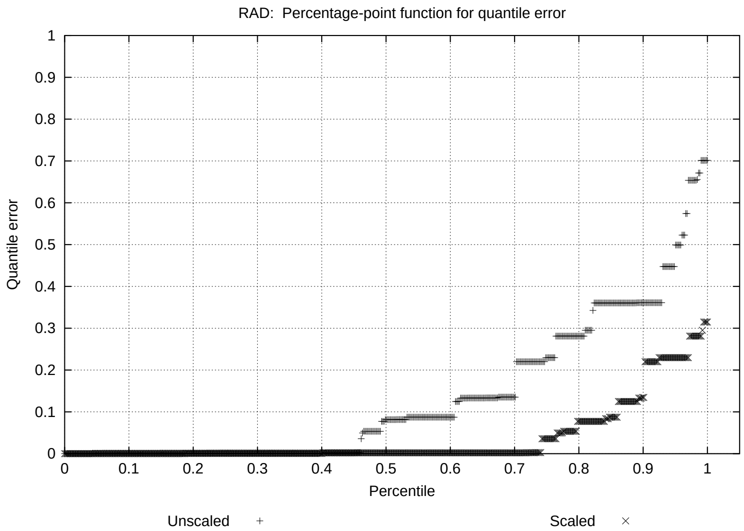


Figure 5.28: Percentage-point curves for quantile error in the unscaled and scaled cases for `RAD` from `housing`. `RAD` has few discrete values, hence the step function. The curves show that the scaled model has much lower quantile errors than the unscaled model.

the alkaline phosphotase concentration; `sgpt`, same for alamine aminotransferase; `sgot`, aspartate aminotransferase; and `gammagt`, gamma-glutamyl transpeptidase – contain continuous values. The sixth attribute, `drinks`, was originally a numerical attribute with steps of 0.5, so it was multiplied by two and treated as an integer. The seventh attribute, `selector`, is an attribute with only two possible values – 1 or 2 – to be predicted and was coded as an integer output, since the system does not currently support unordered (nominal) outputs. For `selector`, 145 tuples registered 1 and the remaining 200 had a value of 2.

Without nonlinear axis scaling Absent scaling, the system chose to retain `alkphos` and `gammagt`. Table 5.11 presents basic results for the remaining five attributes. Quantile results for `selector` would be misleadingly low, as the quantile metric was designed for many values rather than two different values with adjacent quantile ranges; however, classification accuracy is summarized in Appendix F. With regards to the other attributes, the next best performance appears to be on `drinks`, with median and maximum quantile errors of 0.15262 and 0.60320, respectively; for the worst of `mcv`, these are 0.22149 and 0.69551.

Attribute	**Bit cost**	$\sum$ **Q error**^{2}
`mcv`	N/A	24.462
`sgpt`	N/A	21.109
`sgot`	N/A	21.112
`drinks`	2,870	13.409
`selector`	1,126	N/A

Table 5.11: Errors for the `liver` set, without nonlinear scaling. All trees were single-node, as would be expected from the set size and the tree pruning algorithm.

With nonlinear axis scaling With nonlinear scaling, the selector chooses three inputs. These are `alkphos`, with a mixture of normals; `sgpt`, scaled with a single lognormal; and `gammagt`, which received a quarter-power Box-Cox transformation and scaling via the cumulative distribution function of a gamma distribution.

Table 5.12 presents the main results for the remaining four attributes. Classification results again are listed in Appendix F. The sum-of-squared quantile scores for the other attributes also suggest not much of a change. For `drinks`, the median quantile error has dropped slightly to 0.13517 from 0.15262, and the maximum has actually increased to 0.67449, although only one tuple has a quantile error of above 0.5596. The median and maximum quantile errors for `mcv` are still poor at 0.22095 and 0.62213 as well.

Attribute	**Bit cost**	$\sum$ **Q error**^{2}
`mcv`	N/A	23.564
`sgot`	N/A	15.675
`drinks`	2975	12.920
`selector`	1,156	N/A

Table 5.12: Errors for the `liver` set, with nonlinear scaling. All trees were single-node. Comparing with 5.11, the models for `mcv` and `drinks` have not significantly different results but the scaling has allowed a more accurate `sgot` model.

Distribution of errors On the `liver` set, then, we see only marginal improvement from scaling. This is borne out by the percentage-point graphs for the quantile-error statistic, such as Figure 5.29. The curves shown there are unusual in that while the scaled model does better for most cases, the unscaled model actually delivers better results in the percentiles range of 0.1 to 0.2.

5.3.8 Page-blocks

The `page-blocks` data is another data set from the UCI Repository [Murphy and Aha, 1994], and is attributed there to a Donato Malerba at the University of Bari [Esposito et al., 1994].

This set has 5,473 tuples, each of which corresponds to a block from a segmented document. In the original version, each tuple has values for eleven attributes including a five-valued class attribute. In lieu of explicitly adding classification capability to the regression tree system, I expanded this class attribute into five integer attributes, each of which was valued at 0 or 1 depending on whether the tuple had been assigned to that class. This seemed appropriate as the classes have no ordering among them and therefore directly using regression in a single class attribute would make no sense. Thus, this version has fifteen attributes including five integer outputs. Six of the inputs are integers; these include the `height`, `length` and `area` of the block as well as three statistics – `blackpix`, the number of black pixels; `blackand`, the number of black pixels after smoothing; and `wb_trans`, the number of white-black

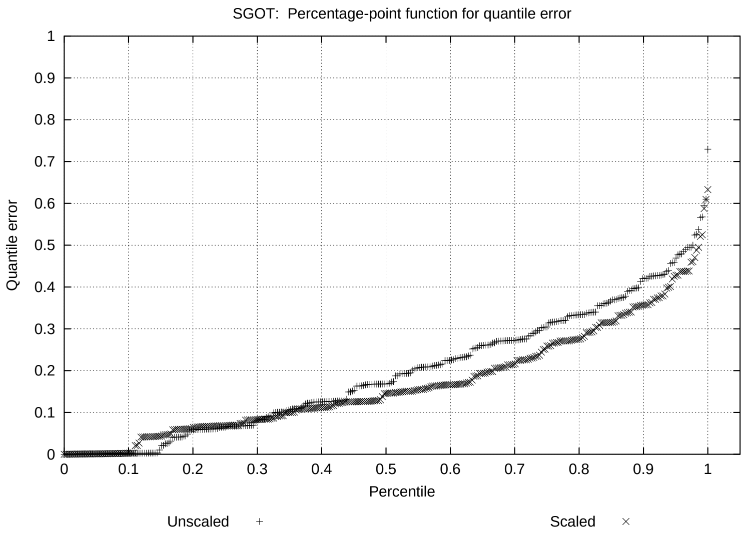


Figure 5.29: Percentage-point curves for quantile error in the unscaled and scaled cases for SGOT in the liver set. The curves diverge somewhat, reflecting the moderate improvements allowed by scaling.

transitions. The remaining inputs – eccen, the eccentricity of the block; p_black, the percentage of black pixels; p_and, the percentage of black pixels after smoothing; and mean_tr, the mean number of white-black transitions – are all continuous.

Without nonlinear axis scaling Absent nonlinear scaling, only two attributes were retained – p_black, the percentage of black pixels with that block; and p_and, the percentage of black pixels after a application of a particular smoothing algorithm. Table 5.13 describes the general results for the modeled attributes in this unscaled case, except for classification results which may be found in Appendix F.

Unusually large errors Regarding the other attributes, logs show that even in the best case of wb_trans, the median and maximum quantile errors were rather high – 0.17379 and 1.2673, respectively. For the similarly scoring height model, these scores were 6.4327×10^{-2} and 0.91210, respectively; there were more quantile errors close to the maximum in the latter case, while in the former case all but two of the quantile errors were at 0.82695 or below. The next best case of length ranges has median and maximum quantile errors of 0.20395 and 0.84192. With blackpix, the model with the worst sum-of-squared quantile error, the median quantile error is 0.21217 and the top 142 quantile errors range from 1 to 9.2147, although all but the top are at 7.4292 or below. As with the `housing` data, while these tuples may be of interest due to their extremely bad modeling the utter lack of annotation makes it fruitless to speculate on possible explanations. `blackand` also generated fifty quantile errors ranging from 1 to 3.695, and had an even higher median error of 0.21354. The highest median quantile error of 0.22582 came from the `area` model.

Attribute	**Bit cost**	$\sum$ **Q error**^{2}	**Nodes**
`height`	31,949	4.7900×10^2	3
`length`	77,035	5.2081×10^2	1
`area`	116,170	6.0553×10^2	3
`eccen`	N/A	7.1550×10^2	19
`mean_tr`	N/A	6.1628×10^2	3
`blackpix`	98,668	2.0657×10^3	11
`blackand`	111,803	8.9800×10^2	11
`wb_trans`	79,427	4.6687×10^2	7
`class1`	7,186	N/A	1
`class2`	6,829	N/A	1
`class3`	5,926	N/A	1
`class4`	6,106	N/A	1
`class5`	6,458	N/A	3

Table 5.13: Errors for the `page-blocks` set, without nonlinear scaling. The five-class `class` attribute from the original set has been broken down into five binary attributes.

With nonlinear axis scaling In the scaled case, the attribute selector retained `p_and`, with a severely asymmetric truncated normal; `mean_tr`, with a $\frac{1}{2}$-power Box-Cox transformation and a mixture of normals; and `blackand`, with a $\frac{1}{3}$-power Box-Cox transformation and another asymmetric truncated normal.

Table 5.14 summarizes the non-classification results after the application of nonlinear scaling; classification results are relegated to Appendix F.

The median quantile errors have dropped dramatically; `area` is now extremely precise at 1.0965×10^{-3}, `blackpix` and `wb_trans` have median quantile errors of approximately 1.8458×10^{-2} and 1.7361×10^{-2}, `length` is not much worse at $3.3443 \times$

10^{-2}, and even the worst median quantile error found via `height` is 0.12555.

Attribute	Bit cost	$\sum$ Q error^2	Nodes
height	29,916	3.9885×10^2	1
length	52,420	53.841	5
area	43,031	1.2202	37
eccen	N/A	2.1420×10^2	5
p_black	N/A	1.3170×10^2	7
blackpix	57,171	9.0255	3
wb_trans	47,328	10.824	11
class1	7,521	N/A	3
class2	7,248	N/A	3
class3	6,671	N/A	5
class4	6,525	N/A	3
class5	6,330	N/A	1

Table 5.14: Errors for the `page-blocks` set, with nonlinear scaling. Sum-of-squared quantile attributes have improved, in some cases extremely – for instance, compare `area`, `blackpix`, `length` and `wb_trans` here and in Table 5.13.

Anomaly: `area` There's an anomaly in the `page-blocks` set. With most of the data sets and their attributes, the regression trees for the same attribute – when an attribute was not chosen for use as an input regardless of the use of the nonlinear scaling module – the trees do not differ greatly in size. The tree for the scaled version of `area` has far more nodes than the tree for the unscaled version. The extra nodes do appear to benefit modeling, however; accuracy on `area` has gone from useless to extremely precise.

Distribution of errors Among the notable percentage-point curves for the quantile error are those for `area` and `eccen`, as seen in Figures 5.30 and 5.31. Figure 5.30 shows a dramatic improvement in the modeling of `area`. In the unscaled case, only `p_and` and `p_black` are retained as inputs; one cannot derive `area` from these regardless of the model class. Even if all attributes but `area` were selected as inputs, in which case sufficient information would be available as $\texttt{area} = \frac{\texttt{blackand}}{\texttt{p\_and}}$, the piecewise-linear nature of the model class combined with a lack of scaling might have led to an even larger model than was necessary in the scaled case. Figure 5.31

shows that `eccen` is one of those attributes for which neither the scaled nor unscaled model provides completely satisfactory performance; still, at least the scaled model dominates here.

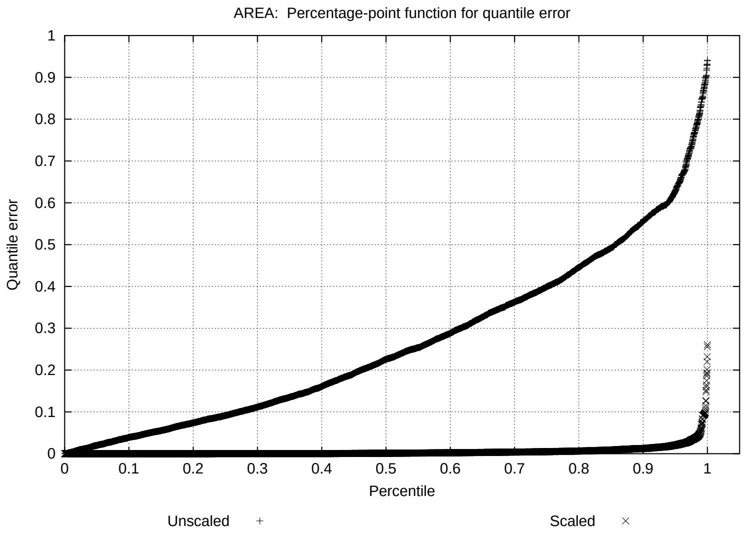


Figure 5.30: Percentage-point curves for quantile error in the unscaled and scaled cases for `area` in the `page-blocks` set. The curve for the scaled model, nearly coincident with the `percentile` axis for most of its length, shows extreme accuracy whereas the curve for the unscaled model shows the opposite.

Bit costs This is one of the few sets used which has quite a few integer attributes, some of which serve as outputs in both the scaled and unscaled cases. Thus, we have multiple bit cost scores which can be compared, and which by definition take into account both the storage costs required for the models themselves and the bits required for a particular error correction scheme that permits perfect reconstruction of the target attribute. It should be noted that the differing metrics do involve different trees, as the MDL-inspired bit cost metric does not take into account skew. Even so, it seems that nonlinear scaling can affect it positively.

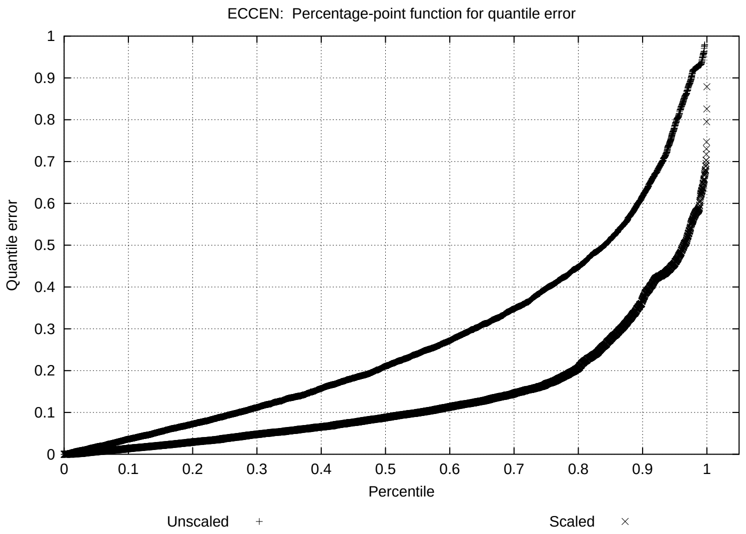


Figure 5.31: Percentage-point curves for quantile error in the unscaled and scaled cases for `eccen` in the `page-blocks` set. The lower curve for the scaled model reflects its improved accuracy.

5.3.9 Wind

The `wind` data set contains average wind speeds measured at twelve meteorological stations in Ireland [Haslett and Raftery, 1989]. Specifically, the data set contains 6,574 tuples, each of which has fifteen attributes. The first three attributes encode the `year`, `month` and `day` in integer format. I set the metadata to ignore the `year` and `day` attributes, but to leave the `month` attribute as a mandatory input to account for seasonal factors. The remaining twelve attributes are continuous and encode average wind speeds at the stations denoted `RPT`, `VAL`, `ROS`, `KIL`, `SHA`, `BIR`, `DUB`, `CLA`, `MUL`, `CLO`, `BEL`, and `MAL`.

Without nonlinear axis scaling Of these, the system keeps `ROS` and `CLA`, as well as the mandatory input of `month`.

Table 5.15 provides the main quantitative results for the modeled attributes. No bit costs are provided as all of the modeled attributes are continuous and therefore the usual prefix-free codes do not apply. Median quantile errors range from 7.7777×10^{-2} for `BIR` to 0.11698 for `MAL`.

Attribute	$\sum$ Q error^2	Nodes
RPT	1.7934×10^2	1
VAL	1.9154×10^2	1
KIL	1.3538×10^2	1
SHA	1.3409×10^2	1
BIR	1.1226×10^2	1
DUB	1.9003×10^2	1
MUL	1.3232×10^2	1
CLO	1.2511×10^2	1
BEL	1.3918×10^2	1
MAL	2.2865×10^2	5

Table 5.15: Errors for the `wind` set, without nonlinear scaling. All modeled attributes were continuous, so no bit costs are listed. The single-node trees may reflect a difficult set, as there are more than enough tuples for the pruning algorithm to allow splits.

Possible outliers Examining the error logs suggest that a couple of tuples may be difficult for the `BIR` model; the quantile errors on November 20, 1964 and November 1, 1968 for `BIR` are both approximately 0.6276, while the remainder of the top 15 errors for that model range from 0.45069 to 0.53956. For `VAL`, the top three quantile errors of magnitudes ranging from 0.83021 to 0.91693 fall on March 18, 1969; December 11, 1963; and February 6, 1966; the next highest quantile error for `VAL` is 0.71662.

With nonlinear axis scaling With scaling enabled, the selector chooses four inputs – `month`, `ROS`, `SHA` and `BEL`.

Table 5.16 lists the main results. Some of the models have significantly lower sum-of-squared quantile errors, notably `RPT`, `VAL`, `KIL`; while others have smaller improvements. Only one attributed modeled in both cases yielded a worse total score, `CLO`, and the degradation here was insignificant. Median quantile errors ranged from a low of 6.2893×10^{-2} for `BIR` and a similar 6.5026×10^{-2} for `CLA`, to two similar high medians of 0.11203 and 0.11275 for `DUB` and `MAL`, respectively. The remaining median quantile errors range from 7.3948×10^{-2} for `KIL` to a cluster of the four remaining models at from 8.3733×10^{-2} to 8.8094×10^{-2}. These median errors are generally close to those in the unscaled case.

Attribute	$\sum$ Q error^2	Nodes
RPT	1.1907×10^2	1
VAL	1.2791×10^2	3
KIL	94.551	1
BIR	79.689	3
DUB	1.7620×10^2	1
CLA	77.873	1
MUL	1.2202×10^2	1
CLO	1.3054×10^2	1
MAL	1.9771×10^2	5

Table 5.16: Errors for the `wind` set, with nonlinear scaling. All modeled attributes were continuous, so no bit costs are listed. Model complexity has increased in two cases, `VAL` and `BIR`; but the sum-of-squared quantile errors drop for not only those but also `RPT`, `KIL`, `DUB` and `MAL`.

Possible outliers Examining the error logs in the scaled case, the occasional tuple may still catch the eye. For `CLO`, the top quantile error of 0.68449 on November 7, 1967 stands out, as the next-highest error is a bit lower at 0.55752. The `MAL` model also has difficulty on a particular tuple – the error for this model reaches 0.81374 on May 3, 1969, while the second-largest quantile error is 0.71975. The separation between worst and ordinary is most pronounced for the `VAL` model; its errors exceed 0.83159 on December 11, 1963 and January 10, 1966, while the next-worst error for `VAL` is 0.56880. It may be of interest that December 11, 1963 is a poor case for `VAL` in the unscaled case, as well. That particular day has a fairly low average wind speed for `VAL`, although it's not the lowest; it was, however, significantly lower than that of the previous day while average wind speeds for nine of the other stations increased.

Distribution of errors For each of the attributes modeled in both cases, nonlinear scaling has made only modest changes to the sum-of-squared quantile error. This is borne out by the percentage-point curves. For instance, the percentage-point comparison graphs for this set range from the closeness of the models for `CLO` as shown in Figure 5.32, to the slight difference for `RPT` as per Figure 5.33.

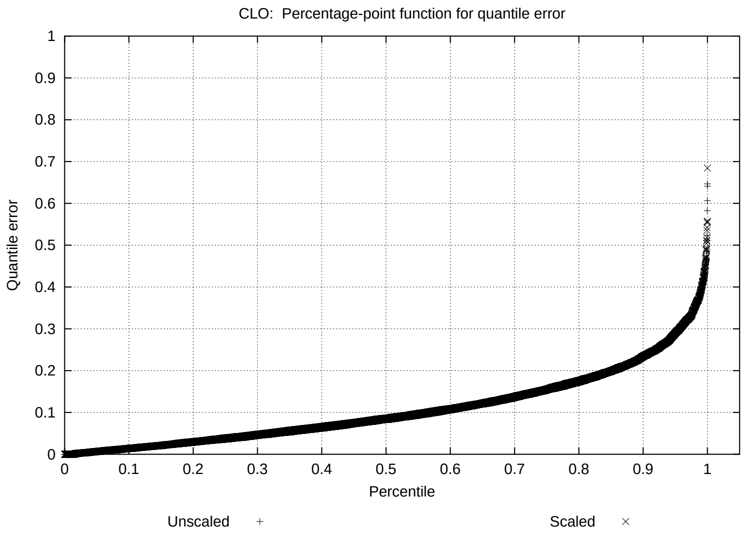


Figure 5.32: Percentage-point curves for quantile error in the unscaled and scaled cases for the `CLO` attribute of the `wind` set. Coincident curves reflect models of nearly identical performance.

5.4 How long did it take?

This is really two questions: how long did it take for each set in total, and how well does the system scale with the cardinality of a data set.

5.4.1 Processing entire sets

Tables 5.17 and 5.18 provide the timing results for the whole data sets. Timing results consist of CPU time results collected via Perl's `Benchmark` package for each individual stage as well as total times. All times should be accurate to the stated figures, and were collected on the same, previously described test platform. For reference, Tables 5.19 and 5.20 specify for each listed data set the number of tuples, total used attributes, and the number of selected or labeled inputs in the unscaled and scaled cases respectively. The latter table also specifies the number of possible outputs; that is, the number of scaled versions of attributes for which no scaled version has been selected as an input.

Timings for the **scaling** stage, where included, include time spent reading the

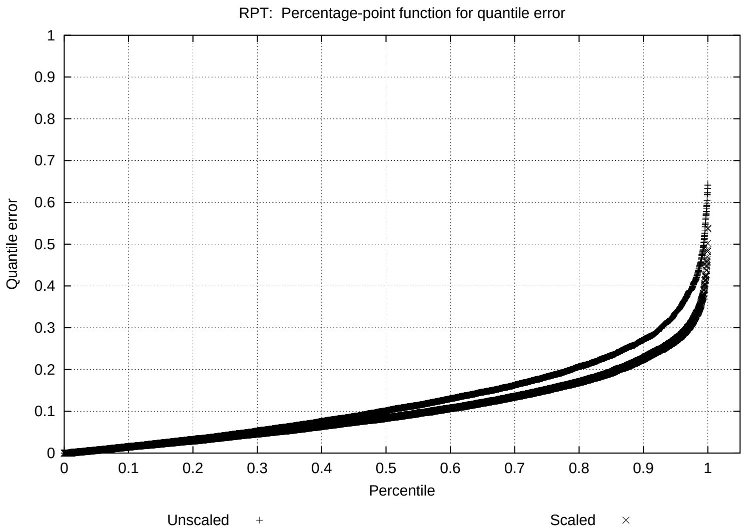


Figure 5.33: Percentage-point curves for quantile error in the unscaled and scaled cases for the `RPT` attribute of the `wind` set. The improvements in `RPT` appear to come mostly for errors worse than the median; the lower curve, corresponding to the scaled model, is coincident with the upper curve elsewhere.

data; applying Box-Cos transformations; estimating parameters for the assorted distributions; applying goodness-of-fit tests; and writing out scaled data and metadata. The **selection** timings include time spent reading the data; iterating through assorted candidate combinations of attributes and computing their fractal dimensions; and writing out the selected data and metadata. Finally, **modeling** results include time spent reading the data; splitting attributes into segments for the purposes of cross-validation; building, pruning and evaluating trees on different subsections of the data; evaluating selected trees on the full data; and recording the trees and their results. For the results below, the class attributes have re-encoded as was described elsewhere, with the exception of `abalone`; while the five-class attribute of `page-blocks` has been expanded to five outputs, and the `liver` set has had `drinks` converted to counting half-drinks rather than drinks, the `rings` attribute of `abalone` retains its original and reasonable 27-value formulation. Input-output determination was also mostly left to the selector, other than an input label for `month` in `wind`; and output labels for `rings` in `abalone`, `wbe`, `wbcw`, `wbhw` in `building`; and the class attributes of `abalone`, `liver`, and `page-blocks`.

5.4.2 Difficulty predicting times

Among the more interesting conclusions, it should be noted that the total time for a set varies quite widely for seemingly similar dimensions. Note that the total time for `page-blocks` greatly exceeds that for `abalone` and `building`. It's also noteworthy that most of the time in both the scaled and unscaled cases the modeling phase dominates, and that the scaling as implemented greatly increases the time spent building or evaluating models. At least part of this must stem from the capability of the nonlinear scaling model to produce multiple versions of each attribute. Since the system builds models for every version of every output attribute, the nonlinear scaling phase as implemented substantially increases the amount of trees that get built. Scaling also appears to encourage a slight increase in the number of attributes used as inputs, which means that each tree involves more work as every computation involves more dimensions. There is also the additional overhead of applying or inverting scaling as need be. The overall effect appears to be, very roughly, a 10-fold increase in computation time. This could presumably be reduced by using stricter goodness-of-fit tests to lessen the currently drastic impact on dimensionality, or even placing arbitrary limits on the number of versions permitted per attribute.

One rough rule of thumb appears to be for any given data set, the use of nonlinear axis scaling increases the absolute amount of time spent modeling in proportion to the increase of the product between the number of inputs and outputs. This is, however, a very rough guideline, and is not useful for prediction across sets.

Data	Selection	Modeling	Total
`abalone`	10.76	5,592.97	5,603.73
`baseball`	5.79	593.71	599.50
`building`	6.52	5,543.15	5,549.67
`CIA World Factbook`	0.07	16.98	17.05
`DJ30`	99.36	7,773.36	7,787.72
`housing`	3.10	763.67	766.77
`liver`	0.91	201.70	202.61
`page blocks`	39.79	10,942.47	10,982.26
`wind`	146.87	19,684.43	19,831.30

Table 5.17: Timing results in the unscaled case. All times are CPU times expressed in seconds. The modeling phase is obviously the most expensive portion, as implemented.

5.4.3 Scalability

If one fixes the data set, it also would be useful to know how well the system scales. This was examined by timing executions of different samples the `page-blocks` set.

For each sample fraction from 10% to 80% at 10% intervals, five independent samples were drawn. These samples were then processed with and without nonlinear axis scaling. Figures 5.34 and 5.35 shows the results for each individual selection, scaling or modeling phase as well as total times with and without nonlinear scaling. In both cases, CPU time required scales linearly with sample size and is dominated by the modeling phase.

This linearity does suggest one approach for determining the feasibility of applying the system to any particular data – perform scaling and selection on the full data, and then build models on samples of a few sizes. If the trend is just as linear as for `page-blocks`, then it should provide a good estimate of the amount of time required for building a model on the full data; in addition, the improvements in accuracy from larger samples, or lack thereof, may be informative as to the value of doing so.

Figures 5.36 and 5.37 show similar tests while sampling over the `DJ30` set. The results without scaling follow the same pattern of minuscule selection time and mod-

Data	Scaling	Selection	Modeling	Total
abalone	529.13	778.04	37,450.88	38,758.05
baseball	359.31	1,365.47	8,142.70	9,867.48
building	625.03	158.60	39,008.61	39,792.24
CIA World Factbook	26.62	1.61	209.87	238.10
DJ30	1,891.80	7,429.93	78,490.96	89,812.69
housing	365.56	264.30	9,746.95	10,376.81
liver	100.54	112.26	1,958.47	2,171.27
page blocks	1,890.11	3,022.71	112,886.66	117,799.48
wind	1,223.67	4,039.61	246,994.80	252,258.08

Table 5.18: Timing results in the scaled case. All times are CPU times expressed in seconds. Again, the modeling phase is extremely expensive compared to the rest. The increased costs versus Table 5.17 likely reflect the greater dimensions resulting from multiple versions of scaled attributes.

Data	# Tuples	# Attr	# Input
abalone	4,177	9	3
baseball	365	17	2
building	4,208	8	3
CIA World Factbook	235	2	1
DJ30	2,529	30	1
housing	506	14	3
liver	345	7	2
page blocks	5,473	11	2
wind	6,574	15	3

Table 5.19: Size of each unscaled data set. Dimensions given relate strictly to the original data less any attributes set to 'ignore'. For unscaled data, the number of outputs is the number of non-ignored attributes less the number of inputs.

Data	Tuples	#Attr	#Scaled	# In	# Out
`abalone`	4,177	9	66	3	45
`baseball`	365	17	253	2	209
`building`	4,208	8	69	3	42
`CIA WF`	235	2	27	1	13
`DJ30`	2,529	30	235	2	208
`housing`	506	14	122	5	95
`liver`	345	7	101	3	44
`page blocks`	5,473	11	125	3	92
`wind`	6,574	15	173	4	117

Table 5.20: Size of each scaled data set. Dimensions given relate strictly to the original data less any attributes set to 'ignore', such as the `year` attribute of the `wind` data. The number of possible outputs is the total number of versions of attributes for which no scaled version was an input; for each such version, trees must be built and tested.

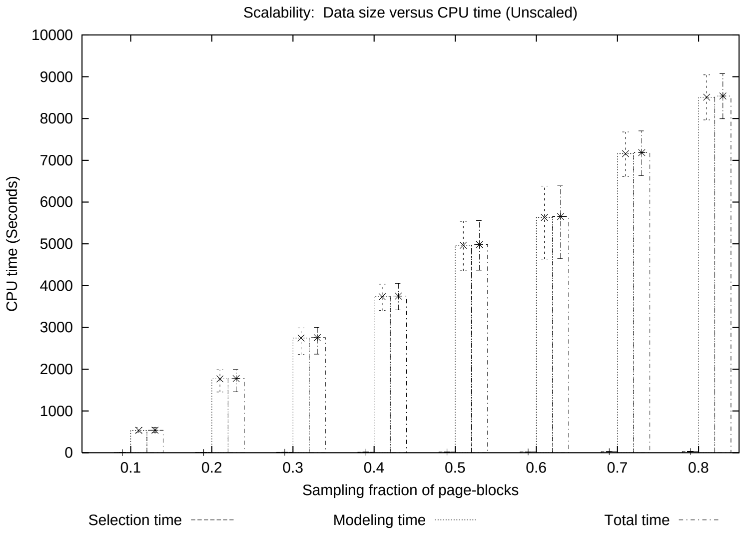


Figure 5.34: Timing runs at different sampling frequencies over the `page-blocks` set, without nonlinear axis scaling. Error bars indicate minimum and maximum timings. For each sampling size, the bars reflect selection time – basically nothing compared to modeling time – modeling time, and total time. The total time appears to scale linearly with sample size.

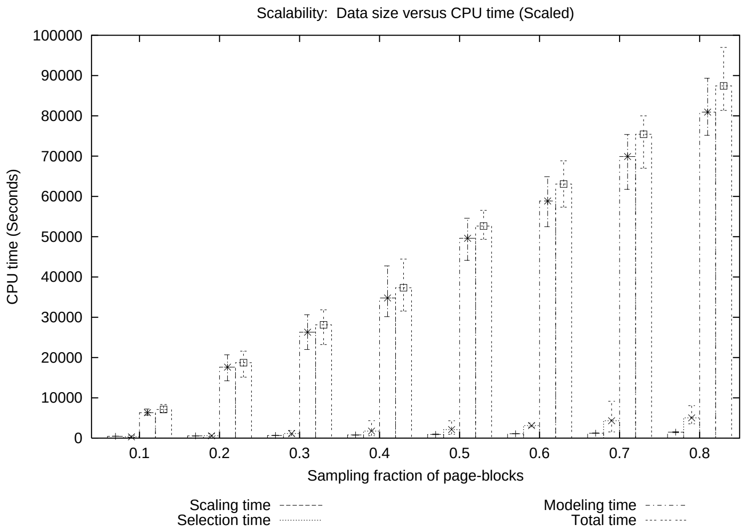


Figure 5.35: Timing runs at different sampling frequencies over the `page-blocks` set, with nonlinear axis scaling. Error bars indicate minimum and maximum timings. Again, individual bars reflect scaling, selection, modeling and total time. The total time shows a linear correlation with sampling size.

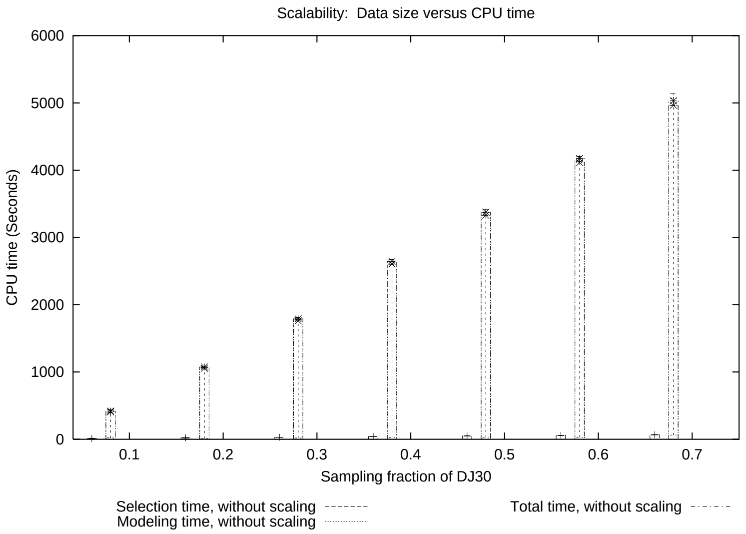


Figure 5.36: Timing runs at different sampling frequencies over the `DJ30` set, without nonlinear axis scaling. Error bars indicate minimum and maximum timings. For each sampling size, the bars reflect selection time – basically nothing compared to modeling time – modeling time, and total time. The total time appears to scale linearly with sample size.

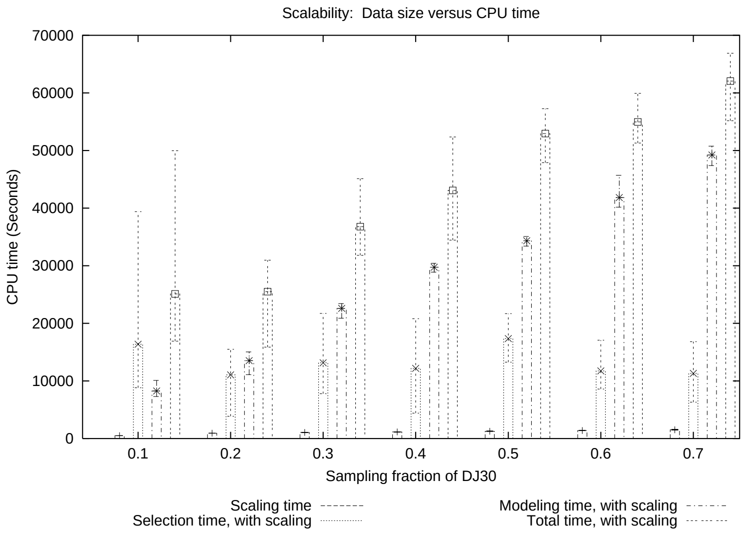


Figure 5.37: Timing runs at different sampling frequencies over the `DJ30` set, with nonlinear axis scaling. Error bars indicate minimum and maximum timings. Again, individual bars reflect scaling, selection, modeling and total time. The total time shows a linear correlation with sampling size.

eling time that scales linearly with the sample size; the results with nonlinear axis scaling are more unstable, but this appears almost entirely due to variance with the selection time. This may be related to the large number of attributes compared to the relatively small amount of data involved. As the sampling size grows, the modeling time continues to grows proportionally; were the set larger, we would expect the modeling time to eventually dominate total time much as it does with `page-blocks`.

Chapter 6

Discussion

Recall that the major concerns of the experiments were the following.

1. What is the impact of nonlinear scaling on the attribute selection process?
2. How good are the resulting models?
3. What else can we find out?

The first two points encompass the major point of this thesis, which was to quantitatively evaluate the impact of a particular automated nonlinear scaling system and its heuristics on the overall behavior of a modeling system. Attribute selection came into play in order to enable dealing with cases in which attributes have not already been labeled as inputs or outputs. The third point deals with the possibility of other information noted either from or about the system.

Results described in the previous chapter lead this investigator to a few basic conclusions. First, nonlinear scaling generally led the attribute selector to select slightly more attributes – although not necessarily an exact superset, nor that many more. Two, model performance as measured by the quantile error often improved significantly, while model complexity remained fairly stable in most cases. Tests with `SBC` and `INTC` in the `DJ30` set suggest that nonlinear scaling can impact model accuracy even when one constrains the unscaled input selection to use the unscaled forms of the inputs selected from the unscaled case. Third, there were some interesting trends such as the dominance of the mixture-of-normals distribution and the odd anomaly. The rest of the chapter goes into greater detail regarding these conclusions.

6.1 Impact on attribute selection

For each continuous or integer attribute in the original set, the system produces another version per enabled Box-Cox transformation. Estimation of each probability distribution followed by Anderson-Darling and χ^2 goodness-of-fit tests for the purposes of filtering poor fits results in the computation of cumulative distribution functions, each of which acts as a scaling function for that particular transformed version. In theory, cumulative distribution functions that result from reasonable distribution fits should reduce skew. These transformed, scaled forms plus the original version of each attribute define the list of attributes which the attribute selector may choose.

Recall that the attribute selector operates on the basis of the fractal dimension. This metric serves as a method for assessing the intrinsic dimensionality of a multidimensional self-similar point-set. While the requirement of self-similarity may impact applicability, it does allow for efficient computation as well as the ability to handle sets that may confound more conventional methods such as those that rely solely on linear components.

The attribute selector greedily selects attributes to maximize fractal dimensionality. By attempting to increase the fractal or intrinsic dimensionality, or equivalently the ideal number of parameters necessary for defining the relevant manifolds, the system should prefer uniformity within each attribute, and independence from attributes it has previously selected. If the scaling system described earlier in this document does actually reduce skew and increase uniformity, we may then expect that attribute selector chooses more attributes. In the experiments conducted, with nonlinear scaling enabled the selector module always chose at least as many attributes as without nonlinear scaling. It did not, however, necessarily choose a superset, nor did it ever choose many more attributes than in the unscaled case. Furthermore, should any improvements in model accuracy be credited to the selection of more inputs, that latter selection can itself be ascribed to the use of nonlinear scaling.

It should also be noted that the selector was always provided with the original version of each attribute in addition to any version whose nonlinear scaling passed goodness-of-fit tests. That only three of the 26 selected inputs from the scaled sets had just the identity transformation strongly suggests that the data was not often uniformly distributed among its axes.

6.2 Impact on the resulting models

While the scaling and selection may itself provide some useful information, the construction and evaluation of regression trees allows the prospective user to evaluate the overall system through the error metric of his choice. In addition, the models themselves may be useful in their own right or the exceptions to them could be of interest; however, the utility of a model or its exceptions does not readily lend itself towards quantitative assessment. Instead, material such as the discovery of explainable anomalies consists primarily of anecdotal evidence and is relegated to the next section of this chapter.

That noted, regarding the performance of the modeling system I have presented three basic types of statistics where applicable. These concern the size of the regression trees, the errors produced by the trees when evaluated against the data sets, and classification accuracy where appropriate. Table 6.1 aggregates high-level results for the first two areas, counting attributes modeled in both the unscaled and scaled cases according to whether the models were more accurate according to the sum-of-squared quantile error, and whether they were more efficient in terms of the number of nodes in the regression trees. The 41 models which either have superior accuracy with the same or fewer nodes, or which have fewer nodes at the same or better accuracy are improvements, modulo the computational complexity of the scaling itself; the five models which are either of inferior accuracy or worse efficiency and without improvements in the other aspect are clearly worse. For four models, neither efficiency nor accuracy has changed much. The 25 remaining models with either better accuracy and worse efficiency or vice-versa would need to be considered more carefully in order to fairly assess their merit.

Classification, while never a main focus of this thesis, is discussed in Appendix F.

6.2.1 Model size

The first class consists of the number of nodes in the selected binary regression tree, from which one may deduce the number of test nodes and leaf nodes. The storage cost of a regression tree as implemented is currently dominated by the leaf node cost. Each test node stores a single attribute index indicating the test attribute and either a single value (for a numerical test; almost all tests were thus) or a set of symbolic values (for a membership test). Every leaf node stores a linear equation containing an additive constant and one coefficient per input attribute. The cost of the tree is

	Better $\sum Q^2$		Same $\sum Q^2$		Worse $\sum Q^2$	
Fewer nodes	**total**	**11**	**total**	**0**	**total**	2
	abalone	1			abalone	1
	DJ30	6			building	1
	housing	1				
	page-blocks	3				
Same # nodes	**total**	**30**	**total**	**4**	**total**	**4**
	baseball	11	liver	2	abalone	1
	CIA-WF	1	wind	2	baseball	2
	DJ30	7			building	1
	housing	6				
	liver	1				
	wind	4				
More nodes	**total**	**23**	**total**	**1**	**total**	**0**
	building	2	abalone	1		
	DJ30	15				
	housing	1				
	page-blocks	3				
	wind	2				

Table 6.1: Tallies for all attributes modeled in both the unscaled and scaled cases. The columns indicate whether or not the version with nonlinear scaling has better, approximately the same (5% difference), or worse results; the rows, whether the model after scaling has fewer, the same, or more nodes than in the unscaled case. The results are shown in both totals and per data set. In most of the cases, scaling helps; this is clearer when one considers that in the case of "more accurate but more complex" models, generally the models are much, much more accurate for not much more complexity.

conditionally independent of the use of nonlinear scaling given the tree structure, the number of attributes selected, and the number of attributes available for selection.

In practice, the majority of trees selected had 11 nodes or fewer, which means 6 or fewer linear equations and 5 or fewer test nodes. Storage costs would therefore be quite small as befits small data sets, on which a larger tree would risk overfitting. Also of note is that scaling as applied rarely had much of an effect on the number of nodes. Any benefit or penalty in modeling accuracy, therefore, could not broadly be attributed to a substantial change in the model complexity other than any additional metadata defining the actual scaling. There were exceptions, such as the `area` attribute in the `page-blocks` set. That attribute received an inaccurate 3-node tree without nonlinear scaling, and a very precise 23-node tree with.

More directly related to tree size than the hypothetical use of nonlinear scaling would be the regression tree building algorithm; in particular, the splitting and pruning criteria. The system as implemented uses a particular set of rules intended to have fairly compact trees; other users may have different opinions on an appropriate balancing point. These decisions, however, were held constant regardless of the use of scaling or the choice of data set and while they introduce the possibility of differing results with other systems, within the scope of these tests the use of nonlinear scaling or the lack thereof is the single major experimental variable.

6.2.2 Errors

Without explicitly presenting the trees, data, source code and per-tuple errors, I have endeavored to present an acceptable description and analysis of how well the models performed in their most fundamental task of regression.

Where attributes have contained only integer values, a cost model has been applied that considers the cost of both the model – using rather basic accounting such as assuming IEEE 64-bit double-precision floating-point numbers for some fields and ordinary prefix-free codes for others – and any additional storage necessary for identifying both the tuple index and the magnitude of any errors. This methodology comes from the semantic compression point of view, and allows one to balance storage cost and model accuracy in accordance with a well-known practice. Few sets used had many integer attributes, however; the two exceptions are `baseball` and `page-blocks`. Noting the limited basis for comparison, it still seems fair to suggest that nonlinear scaling has generally reduced the bit costs, particularly so on the `page-blocks` set on attributes such as `area`.

With continuous attributes, the task of fairly measuring model performance becomes considerably more complicated. I have presented a quantile-based metric designed to look solely at model accuracy from the point of view of a disambiguation task, and which therefore varies expected accuracy with the local data density. Greater precision is necessary in ranges in which more tuples fall, and less for sparser regions.

The previous chapter presents the sum-of-squared quantile errors. Results on the real data have also included median and maximum magnitudes of quantile errors, and the occasional note as to the possible cause of apparent anomalies. In general, the reported quantile errors are better with nonlinear scaling than without, and any increase in the tree size has usually been fairly limited. The occasional exception such as the `length` attribute in the `page-blocks` set shows itself.

6.3 Other findings

Here, we discuss other findings both from and about the system. The former has already been hinted at in terms of investigation about anomalies and the use of different metrics. For instance, the models revealed multiple unusual specimens in the `abalone` set. Other anomalies included a `page-blocks` tuple pertaining to an unusually large block with very few white-black transitions, and possibly explicable errors in the `DJ30` set. For instance, the `DJ30` set provides a case where anomalies in an input attribute might be suspected due to sudden jumps in errors across a wide variety of models. Likewise, for some of its models there exist substantial blocks of time which result much worse error compared to the usual, and for which the nature of the data lets us speculate on a cause. The `World Factbook` area and population data also provides some examples; while with scaling, there exists at least a broad but simple trend between area and population, the model and quantile error allows one to highlight territories and nations that break the trend – such as Greenland, which has a remarkably low population density if you separate it from Denmark.

Evidence from the `baseball` and `page-blocks` sets where integer attributes are common suggests that while there is no clear method that relates quantile errors to a more traditional bit cost system, that there does seem to be relationship as there should be. When scaling aids modeling generation and selection, it seemed to do so for both a traditional integer-only bit-cost scheme and a more general quantile-based error metric.

About the system itself, running numerous experiments has shown a number of interesting facts. For instance, the most commonly chosen probability distribution was a mixture of normals; through its use of multiple components it was significantly more flexible than anything else I included other than the extremely difficult-to-estimate mixture of independently doubly-truncated normals. Truncated normals, single normals, uniform distributions, and the gamma distribution also were selected, but less frequently.

The fractal dimension selector may have been set to be too strict; more attributes would likely have reduced quantile errors further, although in some cases such as `RAD` in the `housing` set and `blackpix` of `page-blocks` set the modeling was already extremely accurate.

A separate issue from correctness is performance. Timing tests suggest that in the current implementation, the modeling phase requires by far the most CPU time of any segment in the system, and that nonlinear scaling greatly increased the time spent there. Any effort spent on improving speed would be best directed at this phase. The timing results also suggest that merely knowing the dimensions of the original data set would not suffice to accurately estimate the required time; note, for instance, the drastic difference in required time for the similarly sized `abalone`, `page-blocks` and `wind` sets. In addition, the CPU time required appears to scale linearly with the number of tuples in the data set, at least on samples of `page-blocks`, in both the scaled and unscaled cases.

Chapter 7

Conclusions

This thesis involved the design, implementation, and examination of a system for the automatic discovery of patterns – functions relating attributes to each other – and their outliers. This encompasses the nonlinear scaling of axes; the labeling of attributes as either inputs or outputs; and the modeling of the mathematical relationships between inputs and outputs, all in an automated fashion with no per-set tuning. Chapter 5 describes the experiments used to analyze the performance of this system in terms of both the accuracy and complexity of the resulting models, as well as the scalability of the system, and Chapter 6 summarizes the results.

As Table 6.1 makes clear, the application of nonlinear axis scaling has indeed helped. The use of scaling enabled more accurate models with the same or less number of nodes in the regression tree 41 out of 75 times; another 23 times, the use of scaling resulted in a more accurate but larger regression tree. In two cases, scaling resulted in the opposite trade-off of a less accurate but smaller model. Four attributes resulted in ties, with trees of the same size and essentially unchanged accuracy. Only in five cases did the system do worse in either accuracy or complexity with no improvement in the other.

The generation of more accurate models also revealed outliers – the exceptions to the otherwise accurate rules. Recall, for instance, Greenland in the `CIA World Factbook` data. It's very sparsely populated, and thus *is* unusual against the usual relationship between area and population – as is obvious with nonlinear axis scaling. Without the scaling, the model for `area` performs much worse and the worst errors appear to be concentrated only in the tiniest regions, regardless of whether their population density is unusual or not. Likewise, axis scaling certainly helped the models in `DJ30`, which in turn revealed a variety of periods of anomalous behavior in individual equities; anomalous behavior that, armed with dates and a search engine, could frequently be linked to events that might plausibly influence the prices in question and cause an individual stock to break away from the usual patterns. With inaccurate models, too many tuples look poor to justify searching for exceptions; but nonlinear axis scaling, even when automated without domain knowledge or data-specific tuning, can improve those models to the point that interesting exceptions can emerge.

In short, nonlinear axis scaling can help – a lot. It's not *guaranteed* to do so, and it adds significantly to computational cost, but it's a powerful tool and one worth serious consideration in addition to simpler methods such as affine transformations solely for normalization.

Appendix A

Doubly-truncated normals

As described in the main body, there exists a long-known estimator for the normal with zero, one or two independent truncation points using the number, extrema, mean, second moment, and standard deviation of the observations [Cohen Jr., 1950]. It requires the simultaneous solution of two nonlinear equations, both of which involve integration, and all of which requires great care when judging convergence or lack thereof.

Perhaps more of a result of implementation-specific decisions rather than any flaw in the published algorithm, this method did not seem to perform especially well relative to the computational cost. Sample results are included below.

Thus, this experimenter designed and implemented an estimator based on a Levenberg-Marquardt numerical search. This estimator first extracts a number of quantiles – either 500 or the number of observations, whichever is less – and the corresponding percentiles. This numerical search operates over the parameter space defined by the unknown mean and standard deviation of the untruncated normal, while trying to minimize the sum of squared error between the observed quantiles and those predicted. The remaining parameters, the two truncation points, are solved for under the assumption that the observed extrema are each above or below $\frac{1}{n}$ of the data where n is the number of observations. Truncation points which would truncate 0.1% or less of the original untruncated distribution are considered nonexistent; that is, the distribution is not truncated in that direction.

To bootstrap estimation, the search begins with the mean and standard deviation of the quantiles. It runs until it reaches a specified number of iterations, it achieves a particular error tolerance, or the sum-of-squared error fails to improve at least a certain amount, whichever comes first; it then will randomly select either its current or its best-known state, possibly alter either or both parameters by up to 10%; and resume searching, for a total of ten executions. The estimator then uses the best-known parameter set. The randomness, while depending on arbitrary parameters, should in theory reduce the odds of returning a poor result due to converging on a highly suboptimal local minimum.

For comparison, I generated a set of 5000 points drawn independently from the 60% to 90% percentiles of a standard unit normal. These truncation points are approximately 0.2526 and 1.2821 original standard deviations above the original mean.

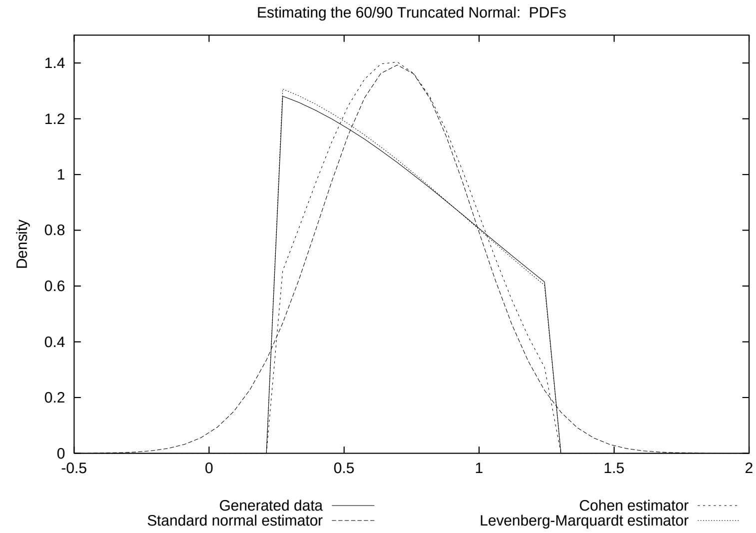


Figure A.1: This figure shows the probability distribution functions (PDFs) for both test and estimated distributions. In this case, the test distribution is a normal truncated at the 60% and 90% percentiles. Here, the Cohen estimator gives very similar results to that of the standard method-of-moments for an untruncated normal, while the Levenberg-Marquardt estimated PDF is essentially coincident with the test distribution.

Figure A.1 shows the estimation results. An ordinary method-of-moments estimator for the non-truncated distribution yields a mean of 0.6960 and a standard deviation of 0.2864 in approximately 0.1 seconds. The moments-based estimator of [Cohen Jr., 1950] consumed 24.8 seconds, and estimated a truncated normal based on an untruncated normal with mean 0.6753, standard deviation 0.3255; it placed the truncation points 1.298 original standard deviations below and 1.863 deviations above the original mean. The numerical search described above consumed 5.8 seconds, and estimated the original untruncated mean to be 4.307×10^{-3} with a standard deviation of 0.9716; repeated executions showed a number of other local minimal with similar PDFs. The truncation points were estimated to be 0.2645 and 1.3238 original standard deviations above the original mean; while not the same as the original generation parameters, this is considerably closer than the moments-based method.

To test a case with truncation points on either side of the mean, I generated another set. Again, 5000 points; this time, the truncation points were set at the 20% and 70% percentiles, or approximately 0.8403 original standard deviations below and 0.5231 above the original untruncated mean.

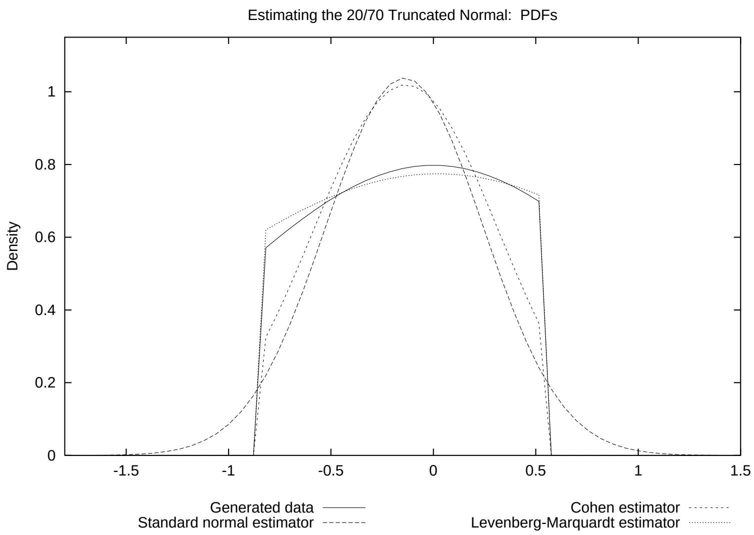


Figure A.2: This figure shows the probability distribution functions (PDFs) for both test and estimated distributions. In this case, the test distribution is a normal truncated at the 20% and 70% percentiles. Again, the Cohen estimator gives similar results to that of the standard method-of-moments for an untruncated normal, while the Levenberg-Marquardt estimated PDF matches the test distribution very well.

Figure A.2 shows the estimation results for this test. The standard estimator for the untruncated normal took approximately 0.13 seconds to estimate a mean of -0.1412 and a standard deviation of 0.3843. The moments-based estimator for the doubly-truncated normal of [Cohen Jr., 1950] required 33.9 seconds to estimate untruncated mean and standard deviation of -0.1343 and 0.4521 respectively, setting truncation points 1.555 and 1.448 original standard deviations below and above the mean, respectively. The search-based method took 5.64 seconds to yield the closest estimate: mean of 2.019×10^{-2}, standard deviation of 1.257, and truncation points 0.6850 and 0.4003 original standard deviations below and above the untruncated mean.

Finally, I generated another 5000-point set, again based on a standard unit normal but with truncation points 1.282 standard deviations below and above the mean, corresponding to approximately the 10% and 90% percentiles. The standard normal estimator took 0.14 seconds to estimate a mean of 2.370×10^{-3} and a standard deviation of 0.6537. After 43.68 seconds, the moments-based estimator suggested an original mean and standard deviation of 3.217×10^{-3} and 0.7483 respectively, with truncation points 1.7174 and 1.708 original standard deviations below and above the untruncated mean. Last, the search-based method took 4.38 seconds to estimate untruncated mean and standard deviation of 5.695×10^{-3} and 0.9665 respectively, with truncation points 1.333 and 1.320 original standard deviations below and above the untruncated mean.

Thus, while the search-based algorithm lacks a certain elegance possessed by analytical methods, it seems to work well on the three types of cases described above – a highly asymmetric truncation with both truncation points above the mean, a middle case with truncation points on either side but asymmetrically placed, and a simple case with truncation points equally far from the mean.

With respect to applicability, it may be noted that of the 1,171 axis scalings over nine data sets that survived the goodness-of-fit tests in the axis-scaling phase, 237 involved a truncated normal. This made it the second-most common scaling, behind 564 uses of the mixture of normals and ahead of 126 invocations of the single non-truncated normal.

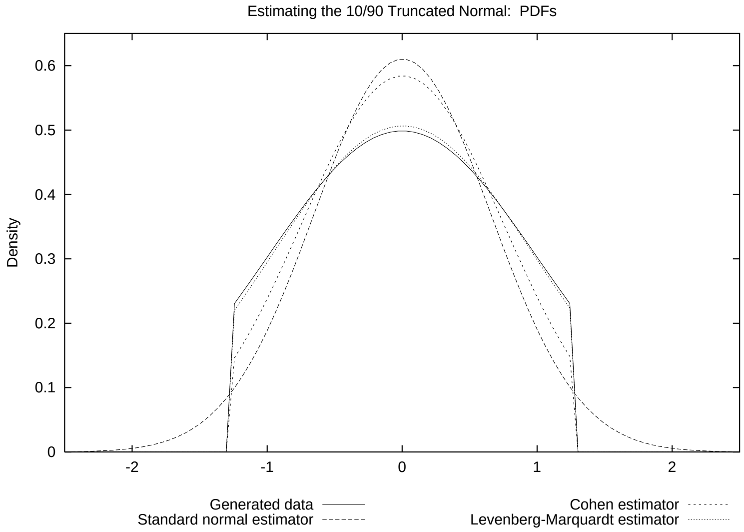


Figure A.3: This figure shows the probability distribution functions (PDFs) for both test and estimated distributions. In this case, the test distribution is a normal truncated at the 10% and 90% percentiles. The Cohen estimator and the untruncated normal estimator both select significantly smaller standard deviations than the test data, resulting in more density assigned towards the mean; the Levenberg-Marquardt estimator again tracks the test distribution very well.

Appendix B

Mixtures of truncated normals

Mixtures of univariate normals may be estimated through a variety of means, primarily expectation-maximization (EM) as described in [Dempster et al., 1977] and [Johnson and Kotz, 1970], and a variation thereof as per [Bradley et al., 1999].

Expectation-maximization for a multicomponent distribution can be summarized as the following procedure.

1. Identify the number of components k. In practice, this may be expensively considered by repeating the rest of the algorithm with different numbers of components and determining which is best according to a suitable metric. The Bayesian information criterion may be considered as a way to balance the complexity of additional components with improvements in log-likelihood.

2. For each component $i = 1..k$, estimate the initial parameters Θ_i with corresponding probability density function f_i, and the component's mixture probability p_i; the sum of the mixture probabilities must sum to 1.

   EM frameworks may leave this as an exercise to the reader, and it can be helpful to adopt a policy of random initialization and multiple executions.

3. Then repeat the following loop until convergence according to some metric such as log-likelihood:

   (a) For value x_j, $j = 1..n$ with n the set cardinality, and each of the k components, compute non-normalized weights $w_{i,j} = p_i f_i(x_j)$.

   (b) Then compute $W_{i,j} = \frac{w_{i,j}}{\sum_{l=1}^{k} w_{l,j}}$. These are the normalized responsibilities that component i has for datum j.

(c) For each component $i = 1..k$, re-estimate Θ_i and the corresponding f_i using the weighted data, and recompute $p_i = \frac{\sum_{j=1}^{n} W_{i,j}}{n}$.

4. Return the best known set of p_i and Θ_i.

This works well for mixtures of ordinary normals. However, it does not directly apply to normals with independent truncation points unless one knows how to estimate the truncation points of the individual components. If a component has non-zero responsibility $W_{i,j}$ for some datum due to that datum being inside the truncation points, it would be up to the component estimator to decide whether to truncate the point by moving the truncation point inwards. Two methods were explored during the course of this work; neither is satisfactory at this stage, but perhaps one will provide a basis for a more reasonable method.

B.1 Modified EM

Perhaps the most obvious approach is to adjust the $W_{i,j}$ estimation. For instance, for any component i one can compute the narrowest range $[\alpha, \beta]$ such that

$$x_j in\, [\alpha, \beta] \rightarrow W_{i,j} \geq \max\left(\mathcal{T}, W_{1,j}, ..., W_{k,j}\right)$$

for some responsibility threshold $\mathcal{T}$, such as 0.3. In other words, a component is at least partly responsible for all points in which it is either the most responsible, or has at least $\mathcal{T}$ responsibility, but for no others.

One can then set $W_{i,j} = 0$ for other values of j, set the truncation points accordingly, and re-normalize all $W_{i,j}$ weights for new p_i mixture probabilities after all $W_{i,j}$ truncation has been performed. Choosing $\mathcal{T}$ provides an obvious difficulty, as does undoing the truncation of a point from any component's area of responsibility.

B.2 Windowing

A second approach sacrifices much of the theoretical flexibility by ignoring the possibility of significant overlap between components. This reduction in theory should permit a divide-and-conquer approach while still allowing some distributions that might be ugly to model using a mixture of ordinary normals.

The first priority of the windowing method is to identify potential boundaries between components; let us call these potential boundaries "breakpoints". To do this, the system starts by using a Gaussian kernel to estimate density at every datum. The local minima of the results, or regions of unusually low density, provide the first set of breakpoints as they may correspond to areas between components.

The estimator then applies a natural logarithm to the estimated densities. The logarithms of the density function of a normal have a certain resemblance to a parabola. Parabolas can be modeled via quadratic equations, which conveniently can be iteratively fit using an algorithm called recursive least-squares (RLS). RLS has a parameter which controls the stability of the estimation with respect to updates and shifting trends; therefore, running to RLS estimators on the same data with different stability estimates and checking for divergence may provide a way of estimating when one parabola – or component – begins and another ends. So long as the estimate the same parabola as previously, both models should have similar values; when shifting from one to another they may jump at different points. Therefore, areas of sharp divergence suggest breakpoints, and then the RLS models are reset.

This yields a set of breakpoints which defines a segmentation of the data. The windowing estimator then starts at an extrema, and greedily looks for the largest consecutive set of segments including that extrema which can be plausibly estimated with a single truncated normal according to goodness-of-fit tests. The longest set with a satisfactory fit is then removed, and the estimation continues.

This process is clearly rather computationally expensive and sacrifices flexibility; practice suggests that performance, applicability or both is still far inferior to that of the standard mixture of normals.

Appendix C

Estimating the fractal dimension

Recall that the D_2 fractal dimension is defined as

$$D_2(X) = \frac{\partial \log p_r(X)}{\partial \log r}$$

where

$$p_r(X) = |\{(x,y)\,|x, y \in X \wedge d(x,y) \leq r\}|$$

for an infinite, perfectly self-similar set; under such conditions, the partial derivative is constant over varying r. Finite sets force the use of such techniques as looking for a maximally linear contiguous subset of the $< \log r, \log p_r >$ pairs of a minimum size.

The naive implementation requires checking every possible pair of values, and is clearly quadratic. For data sets of realistic size, this is not a reasonable approach.

Fortunately, there are quite serviceable approximations with better scalability. Among these are the two following methods; while both in practice scale linearly with the data set size, the former likely has a lower constant factor, while the latter requires far less storage and may be performed incrementally.

C.1 Box-counting

The first and more well-known of the two has been called "box-counting" [Schroeder, 1991] [Traina Jr. et al., 2000]. A basic version of this algorithm is described in Figure C.1.

---

1. Define a geometric series of distances $r_i = \alpha\beta^i$ such that $\beta \in \mathcal{Z}^+ \wedge \beta \geq 2$ and $i \in \mathcal{Z}$. The distances r_i should ideally be chosen with some respect to the data set $X \in \mathcal{R}^k$.
2. For each r_i, perform the following steps.
   (a) Define a k-dimensional hypergrid with cells of length r_i along each of the k axes. A simple approach is to compute a coordinate $x_j \in \mathcal{R}$ into an index $\lfloor \frac{x_j}{r_i} \rfloor$.
   (b) Assign every k-attribute tuple into its cell.
   (c) Compute the sum-of-squared cell occupancies $C_{r_i}(X) = \sum c_j^2$.
3. Compute the approximation

$$\hat{D}_2(X) = \frac{\partial \log C_r(X)}{\partial \log r}$$

Figure C.1: The box-counting algorithm.

The algorithm described in [Traina Jr. et al., 2000] improves performance by using a hierarchical data structure to store and compute counts at multiple resolutions r instead of requiring one pass per radius and without simultaneously, explicitly storing $|\{r_i\}|$ hypergrids.

Likewise, one could use a database for storing cell counts, and iterate over occupied cells at one radius r to compute the counts for a coarser radius βr. The computational cost of the basic algorithm and variations thereof may all be expected to scale approximately linearly with the data cardinality, but at a potentially high storage cost depending on how one tracks cell occupancies.

C.2 Tug-of-war

This investigator was among those involved in the work described in [Wong et al., 2003] and [Wong et al., 2005] which focused on developing an algorithm for approximating the fractal dimension that used a minimal amount of storage space with respect to the number of tuples, could be performed incrementally, and still possessed a linear scalability in terms of computational cost versus the number of tuples involved.

The tug-of-war algorithm relied on a result from [Alon et al., 1996], which showed a way to compute the second moment of the frequencies of a set, incrementally and inexpensively, with probably-approximately correct (PAC) bounds. The Tug-of-War algorithm uses this approach to approximate the $C_{r_i} = \sum c_j^2$ second-moments that would otherwise be computed during box-counting.

As noted above, for relevant theory and experimental results see [Alon et al., 1996] [Wong et al., 2003] [Wong et al., 2005]. The above algorithm in practice does return good approximations with minimal storage costs and computational cost linear in the number of tuples, although the constant factor is likely higher due to the amount of calculations. For a large data set that would risk overwhelming box-counting due to excessive storage costs, tug-of-war would be a reasonable choice.

1. Set parameters $s_1, s_2 \in \mathcal{Z}^+$. s_1 reflects the level of approximation, and s_2 the related probability. Setting $\lambda = \sqrt{\frac{16}{s_1}}$ and $\epsilon = e^{-\frac{s_2}{2}}$, the probability that the estimate of the second moment has a greater relative error than λ will be at most ϵ.

2. Generate $s_1 s_2 R$ random four-wise independent hash functions, where $R = |\{r_i\}|$ or the number of radii.

   Each hash function is based on a four-tuple of random, distinct primes $(\pi_1, \pi_2, \pi_3, \pi_4)$ as well as a large and distinct prime q which need not vary per function. Then, for a tuple that would have a k-dimensional hypergrid index of $(g_1, ..., g_k)$ where $g_j = \lfloor \frac{x_j}{r_i} \rfloor$, the hash function returns -1 or 1 depending on the parity of the number of 1s in the value *result* where

$$result = \sum_{j=1}^{k} \pi_1^k g_j^3 + \pi_2 g_j^2 + \pi_3 g_j + \pi_4^j \text{mod} q$$

3. Associate each of the hash functions with a counter initialized at zero.

4. For every tuple, increment or decrement every counter based on the result from the corresponding function.

5. To estimate C_r for a given radius r:

   (a) Partition the $s_1 s_2$ counters into s_2 sets of s_1 counters each.

   (b) Compute the square of each counter.

   (c) Compute the arithmetic mean of the squares in each subset.

   (d) Compute the median of the means. This is the estimate.

6. The D_2 fractal dimension can then be estimated at any time by estimating the C_r values as above, and then performing the usual partial derivative computation.

Figure C.2: The tug-of-war algorithm.

Appendix D

Regression trees

The phase which performs the actual modeling uses a piecewise-linear regression tree based upon the SECRET system described in [Dobra and Gehrke, 2002], but with substantial differences. This appendix provides more details about the construction, pruning and use of the tree.

D.1 Construction

As with SECRET, the modeling phase produces piecewise-linear binary regression trees through separate construction and pruning phases. The current implementation adopts the same procedure of estimating a mixture of two multivariate normals in order to apply pseudoclass labels for the purposes of split-point selection; here, the estimator is the method of [Bradley et al., 1999].

Only axis-aligned splits are currently supported. For these, we again follow the logic of SECRET and rely on either the Gini index, for splitting along discrete attributes; or the method of [Loh and Shih, 1997], for continuous attributes. There may also be differences in certain criteria which limit the growth of the tree, such as not splitting a node which covers only 10 or fewer tuples. The linear models themselves are computed at each node using either recursive least-squares, when dealing with more than 50 tuples; or the singular value decomposition, otherwise.

Otherwise, however, tree construction should be rather similar to the normal SECRET method. Both processes result in binary trees where internal nodes contain tests – whether the value of a continuous attribute is less than or equal to a threshold, or whether the value of a symbolic attribute is within a certain set – and leaf nodes contain linear models based on all data that reached that leaf.

D.2 Pruning

The most significant differences occur here. The published tests with SECRET use Quinlan's resubstitution error pruning as per [Quinlan, 1993].

In addition to cross-validation techniques described in the next section, the version written for this thesis uses a pruning metric that explicitly converts error magnitudes into bits in order to compare the presumably equal or lower error of splitting a node into two with the increased cost of storing the parameters for the two children. To do this, it needs a profile of the likely error distribution at any given node. It would be too expensive to recompute error profiles on the basis of individual prune/split decisions, so instead the system summarizes errors based on the space-efficient approximate quantizer of [Greenwald and Khanna, 2001] to provide profiles of the distribution of error magnitudes. If a split is pruned, the error profile originally computed using the model in the parent is used; if not, it can be estimated from the at-most fifty error magnitudes stored in each profile and the relative population sizes of the children.

To convert error magnitudes into bits, the system first normalizes them by dividing them by the range of the data covered by the node in question. These fractions are then multiplied by 1000 and rounded down, resulting in non-negative integers which can be assessed in terms of standard prefix-free codes. The total cost of an error profile can then be counted in bits and compared against the cost of storing parameters for the relevant subtrees. Setting this balance has obvious issues, but the current setting should prevent pruning from being excessive when the benefits in reduced errors are not trivial compared to the number of inputs and thus the number of parameters required per child.

D.3 Cross-validation

As an additional hedge against overfitting, the model builder performs four-fold cross-validation as follows.

A data set is randomly partitioned into four segments of equal size. The system then runs four independent trials, each of which selects two segments for building the regression tree, one segment for pruning it, and the final segment for evaluating. The tree builder returns the tree which had the lowest sum-of-squared error on its evaluation segment. It would seem reasonable to change this to evaluate on the whole data set, however, and perhaps to use the quantile metric espoused in other parts as well.

D.4 Quick comparison

For comparison, I checked the `abalone` data set from the UCI repository [Murphy and Aha, 1994] using the data as-is with all attributes except `rings` labeled as inputs, and `rings` as the output. This mirrors one of the tests executed in [Dobra and Gehrke, 2002], where numerous trials achieved a mean squared error of 4.76 when modeling `rings` whether or not using oblique splits; the complexity of the resulting tree was unfortunately not published.

In addition to the pruning method described above, I also evaluated a more traditional pruning method that cares only about accuracy, and thus reduces overfitting only due to the use of a holdout set for pruning or selecting among multiple trees. This investigator's code with the 'balanced' and storage-aware pruning above, and the four-fold cross-validation and tree-selection process yielded a single-node tree with a worse prediction error of 5.011; the use of scaling barely reduced this to 4.9461 from a single-node tree. With the accuracy-only pruning method and the cross-validation harness, the model builder selected a 13-node tree which had a mean squared error of 4.602 *sans* scaling. Combining the accuracy-only pruner and the nonlinear scaling, a 25-node tree was obtained with a mean squared error of 4.361 rings.

The `Boston` data set referenced in the evaluation of `SECRET` is the same as the set labeled `housing` elsewhere in this document. Using the same encoding as `SECRET` – original data, all attributes as inputs except the median house value `MEDV` – on which `SECRET`'s piecewise-linear trees with oblique splits yielded a mean squared error of 26.11, and 30.91 when restricted to axis-orthogonal splits. The single-node tree using the balanced pruner had a mean-squared error of 24.50, and with the accuracy-only pruner a seven-node tree was found that had a mean squared error of 15.17 – without using orthogonal splits, since none have been implemented. Combining nonlinear scaling and the accuracy-only pruner, the mean squared error rose to 23.69 from 5 nodes.

Appendix E

Linear models

One obvious question might be "What if we discarded the nominal attributes and built only linear models?". Linear models, while obviously far less versatile than the piecewise-linear trees built in Chapter 5, do have some advantages; these lines are fast to compute, may be readily interpreted in terms of inputs and outputs, and require no judgments regarding pruning or split criteria.

Does nonlinear axis scaling improve the performance of these simpler models? And how well do these models perform, compared to the more versatile but more complex models discussed elsewhere?

E.1 Experiments

To answer these questions, we can build linear models on the same data sets and consider the results. The choice of nonlinear axis scalings and of inputs remains the same, as does the use of the quantile metric. Cross-validation was not used; instead, regression was performed on the full data.

E.1.1 Abalone

Table E.1 shows the results for linear models for the `abalone` set. Comparing with Tables 5.1 and 5.2, it is striking how close in accuracy the post-nonlinear scaling linear models are to the more complicated models described in Chapter 5. The unscaled linear models are more often significantly less accurate than their piecewise-linear counterparts.

Attribute	$\sum$ Q error^2 Unscaled	Scaled
`Length`	(input)	19.854
`Diameter`	*8.7731*	22.106
`Height`	(input)	52.123
`Whole weight`	25.815	*7.5109*
`Shucked weight`	41.032	(input)
`Viscera weight`	39.281	(input)
`Shell weight`	*34.074*	38.348
`Rings`	*87.237*	96.912

Table E.1: Quantile errors for linear models on the `abalone` data set, with and without nonlinear axis scaling. *Emphasized* numbers indicate lower (better) sum-of-squared quantile errors for attributes modeled in both the unscaled and scaled cases.

E.1.2 Baseball

Since the original models described in Table 5.4 are all linear, the table in Table E.2 reflects only the stability of the model-building process. Clearly the use of samples for the purposes of cross-validation in the original model building process generally had little impact on the accuracy compared to building the model on the entire data. The trend of the nonlinear axis scaling improving accuracy is quite clear here.

E.1.3 Building

The accuracy results for the linear models in Table E.3 may be compared with Tables 5.5 and 5.6. For the models which were linear even in the unconstrained case – unscaled `hour`, `wbe` and `wbcw`, scaled `wbcw` and `wbhw` – the model performances reported here are only insignificantly changed. The other models, however, have suffered to varying degrees. The nine-node model for a scaled version of `solar`, for instance, has a sum-of-squared quantile error of only 46.270. The five-node model for the unscaled version of `humid`, however, is not that much of an improvement with a sum-of-squared quantile error of 1.9975×10^2.

Attribute	$\sum$ Q error^2 Unscaled	Scaled
`BA`	28.405	*15.284*
`SLG`	30.052	*16.371*
`OBA`	28.867	(input)
`G`	17.386	*9.0858*
`AB`	16.258	*7.6846*
`R`	16.221	*6.4394*
`H`	16.379	*7.9469*
`TB`	18.936	*6.0344*
`2B`	16.330	*7.8002*
`3B`	*4.1074*	4.5415
`HR`	24.407	*5.8400*
`RBI`	21.511	*6.3525*
`BB`	20.523	*5.1025*
`SO`	19.906	(input)
`SB`	*12.026*	13.390
`CS`	(input)	9.0217
`E`	(input)	12.347

Table E.2: Quantile errors for linear models on the `baseball` data set, with and without nonlinear axis scaling. *Emphasized* text reflects better scores.

Attribute	$\sum$ Q error^2 Unscaled	Scaled
`hour`	2.9825×10^2	(input)
`temp`	(input)	1.7441×10^2
`humid`	2.1083×10^2	(input)
`solar`	6.3164×10^2	*2.9427×10^2*
`wbe`	3.5329×10^2	*2.8951×10^2*
`wbcw`	*74.540*	1.5151×10^2
`wbhw`	*66.743*	1.8206×10^2

Table E.3: Quantile errors for linear models on the `building` data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

E.1.4 CIA World Factbook (2001)

In the unconstrained case, with or without scaling, a single linear model was built. Re-running this model without cross-validation therefore serves only as a stability check, not for evaluating whether or not a more powerful model would have been more useful.

In both the unscaled and scaled cases, the sole model was for `area`. The sum-of-squared quantile errors are 33.991 and 6.0865, for the unscaled and scaled cases, respectively. Interestingly, the score for the unscaled case is quite different from the original score of 16.901; the use of cross-validation, or lack thereof, can clearly make a difference. The scaled score, however, is essentially identical with the original of 6.0886.

E.1.5 DJ30

The sum-of-squared quantile errors for the strictly linear models may be found in Table E.4. As per the piecewise linear models described in Tables 5.7 and 5.8, nonlinear axis scaling improves the accuracy for every attribute modeled in both cases.

Few of the unconstrained models were linear. In the unscaled case, these were `GM` and `SBC`; in the scaled case, `DD`. In all three cases, the sum-of-squared quantile errors are quite similar with and without cross-validation. For the remaining cases, the piecewise linear models are quite frequently significantly better with some of the piecewise linear models reducing the sum-of-squared quantile error by 50% or more versus their linear counterparts.

E.1.6 Housing

Table E.5 shows sum-of-squared quantile errors for linear models on the `housing` data. Table 5.9 shows that of the unscaled unconstrained models, all but `CRIM` and `ZN` were already linear; the linear versions of `CRIM` and `ZN` produce sum-of-squared quantile errors that are approximately three and six times the sum-of-squared quantile errors for the piecewise-linear models. For the remaining models, the sum-of-squared quantile errors have changed little except for `B`; the original model for that set, built only on a sample rather than the full data, performs extremely poorly on a large number of tuples.

Attribute	$\sum$ **Q error**^{2} Unscaled	Scaled
`AA`	1.1546×10^2	*78.020*
`AXP`	1.4554×10^2	*64.860*
`BS`	1.4868×10^2	*1.1393* $\times$ *10*^{2}
`CAT`	69.824	*53.718*
`CI`	1.2675×10^2	*67.078*
`DD`	49.783	*30.857*
`DIS`	1.3179×10^2	*31.149*
`EK`	1.9299×10^2	*1.8490* $\times$ *10*^{2}
`GE`	1.4953×10^2	*56.475*
`GM`	1.2261×10^2	*36.928*
`HD`	1.4023×10^2	*62.095*
`HON`	82.922	*20.819*
`HPQ`	1.3815×10^2	*48.449*
`IBM`	1.7232×10^2	*1.4963* $\times$ *10*^{2}
`INTC`	1.2494×10^2	*38.427*
`JNJ`	1.4938×10^2	*1.2124* $\times$ *10*^{2}
`JPM`	1.0605×10^2	(input)
`KO`	2.0656×10^2	*1.9234* $\times$ *10*^{2}
`MCD`	1.1191×10^2	*31.586*
`MMM`	1.2533×10^2	*1.1757* $\times$ *10*^{2}
`MO`	1.5020×10^2	*1.1180* $\times$ *10*^{2}
`MRK`	1.6696×10^2	*52.027*
`MSFT`	1.2779×10^2	*46.582*
`PG`	99.837	*68.998*
`SBC`	1.1269×10^2	*35.528*
`T`	2.0627×10^2	*1.8909* $\times$ *10*^{2}
`UTX`	1.4751×10^2	*1.0385* $\times$ *10*^{2}
`WMT`	1.7422×10^2	*1.1318* $\times$ *10*^{2}
`XOM`	1.8783×10^2	*99.841*

Table E.4: Quantile errors for linear models on the `DJ30` data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

With respect to the scaled models as per Table 5.10, all the original unconstrained models except for `TAX` were already linear. The linear model for scaled `TAX` performs comparably to the 3-node (single-split) model; the other linear models perform quite similarly to the original linear models built using cross-validation, with deviations of up to approximately 10% in sum-of-squared quantile error.

	$\sum$ Q error^2	
Attribute	**Unscaled**	**Scaled**
`CRIM`	1.1706×10^2	(input)
`ZN`	42.406	*2.7931*
`INDUS`	14.337	*13.096*
`NOX`	(input)	8.7608
`AGE`	2.0387×10^2	(input)
`DIS`	22.230	*12.471*
`RAD`	22.975	*4.2139*
`TAX`	28.401	*16.173*
`PTRATIO`	28.503	*26.861*
`B`	5.1174×10^2	(input)
`LSTAT`	15.732	*12.886*
`MEDV`	21.988	*16.673*

Table E.5: Quantile errors for linear models on the `housing` data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

E.1.7 Liver

Table E.6 shows the results for the strictly linear models for the `liver` set. Tables 5.11 and 5.12 note that the original models were also strictly linear, so this serves as test of sensitivity to sampling versus building a model on the entire data. The differences in sum-of-squared quantile errors between those results and these are approximately 5% or less.

	$\sum$ Q error^2	
Attribute	**Unscaled**	**Scaled**
mcv	24.288	*23.365*
sgpt	20.448	(input)
sgot	20.633	*15.575*
drinks	12.946	*12.448*

Table E.6: Quantile errors for linear models on the liver data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

E.1.8 Page-blocks

Table E.7 shows accuracy results for strictly linear models built using the full page-blocks data, sans cross-validation. These may be compared with the results for the unrestricted, cross-validated models found in Tables 5.13 and 5.14. Among the unconstrained models for these non-class attributes, only length (unscaled) and height (scaled) used linear models; the rest ranged from 3 to a remarkably high 37 nodes.

The sum-of-squared quantile errors has deviated somewhat for the scaled version of height, with the model described here having approximately a 10% lower sum-of-squared quantile error. The deviation for the unscaled variant of length, however, was significantly lower.

We can also examine the performance differences between the multi-node trees built using cross-validation and the linear models built on all the data. For the unscaled outputs, there are large deviations for area, eccen, blackpix, blackand and wb_trans; with area, blackpix and blackand, the multi-node models are actually *worse*. For the scaled outputs, all of the linear models are worse than their multi-node counterparts – extremely so for area, where the use of 37 nodes instead of 1 reduces the sum-of-squared quantile error by a factor of 71.6, and wb_trans for which the 11-node model reduces the sum-of-squared quantile error by a factor of approximately 22.0.

E.1.9 Wind

Table E.8 shows the sum-of-squared quantile errors for the linear models for the wind set. These may be compared with Tables 5.15 and 5.16. Among the original

Attribute	$\sum$ Q error^2 Unscaled	Scaled
`height`	5.7438×10^2	*3.5837×10^2*
`length`	5.2446×10^2	*67.553*
`area`	1.3623×10^3	*87.561*
`eccen`	8.1249×10^2	*2.5596×10^2*
`p_black`	(input)	1.8553×10^2
`mean_tr`	6.7501×10^2	(input)
`blackpix`	1.3003×10^3	*11.252*
`blackand`	7.1588×10^2	(input)
`wb_trans`	5.6019×10^2	*2.2591×10^2*

Table E.7: Quantile errors for linear models on the `page_blocks` data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

unconstrained models for the unscaled outputs, only `MAL` has a multi-node model; for the scaled outputs, multi-node models exist for `VAL`, `BIR`, and `MAL`.

For the unscaled outputs, the linear models produce quite similar results regardless of whether or not they are built on the full data set. The 5-node model for `MAL` has approximately 10% lower sum-of-squared quantile-error than the 1-node model.

Considering the scaled outputs, we see that the linear models built on the full set produce essentially the same sum-of-squared quantile errors as the linear models built using cross-validation and pruning sets, for `RPT`, `KIL`, `DUB`, `CLA`, `MUL` and `CLO`. The 3-node model for scaled `VAL` is barely more accurate; the 3-node model for scaled `BIR` is insignificantly worse; and the 5-node model for scaled `MAL` is again has roughly 10% lower error than the linear model.

E.2 Summary

A few notable conclusions can be drawn from these constrained tests.

First, it is clear that should the model class be constrained to strictly linear regression, the use of nonlinear axis scaling almost always allowed improved model accuracy. This can be true even for sets such as `baseball`, for which the number of

Attribute	$\sum$ **Q error**^{2} Unscaled	Scaled
RPT	1.7886×10^2	*1.1906×10^2*
VAL	1.9118×10^2	*1.3050×10^2*
KIL	1.3592×10^2	*94.548*
SHA	1.3380×10^2	(input)
BIR	1.1179×10^2	*78.384*
DUB	1.8963×10^2	*1.7609×10^2*
CLA	(input)	77.736
MUL	1.3235×10^2	*1.2201×10^2*
CLO	*1.2504×10^2*	1.3026×10^2
BEL	1.3977×10^2	(input)
MAL	2.5007×10^2	*2.1876×10^2*

Table E.8: Quantile errors for linear models on the `wind` data set, with and without nonlinear axis scaling. *Emphasized* scores are the better of the two – unscaled or scaled.

inputs for the models remained unaffected by the use of scaling.

Second, based upon the comparison between these models – built using the full data set, with no cross-validation – and the linear models that sometimes resulted from the full cross-validation process, for many of these attributes there was no significant difference in accuracy – but for a few the differences were quite drastic. It is not clear whether this could be readily predicted outside of actually taking multiple samples, although the presence of extreme outliers relative to the regression results on the full set would presumably be one circumstance that could cause such a gap. In a couple of cases, the model built using cross-validation was actually *worse*, which suggests perhaps a poor choice of sample for the test phase. Using sampling only for the allocation of tuples between building and pruning, but testing on all the data, may be a better choice.

Third, in many cases the multi-node models do significantly better than the strictly linear models – even though the linear models are actually built on all the data while the multi-node models have been built on half the data, pruned on a quarter, and selected from based on evaluations on the last quarter. In extreme cases, the sum-of-squared quantile error from the linear models can be many times that of the multi-node models.

Appendix F

Classification

First, it is important to note that this thesis project by design did not focus on classification. Classification differs strongly from regression, with the most obvious difference being that regression requires accurately predicting along an ordered continuum while classification stipulates a possibly unordered discrete set of outcomes. Regression lends itself to regression trees, but the application of such trees to classification is much more complicated. Should one create multiple trees, one per class, and declare that the prediction matches the tree with the highest answer? In addition, do the heuristics commonly used to build regression trees, such as those designed to minimize the sum-of-squared error, actually make any sense on a classification task? It would be more logical to use a tree built with a heuristic aimed at classification, such as the Gini index or Shannon information gain.

However, the `abalone`, `liver` and `page-blocks` data sets do have specific classification tasks, and it is an obvious question as to how well the regression trees could be made to classify, and how much of an impact if any nonlinear axis scaling had on these tasks. The results are quite mixed; for instance, with `page-blocks` combining the accuracy-only pruner and axis scaling gives a much more efficient solution for the same accuracy as without scaling, but case of `abalone` and `liver` the same comparison leads to the opposite conclusion. Given the impact it *can* have, however, it may be of interest to examine how it can be better applied.

F.1 Abalone

The documentation with the `abalone` set notes that the previous experiments in classification aggregated the 29 distinct values for the `rings` attribute into three classes. After doing so, a backpropagation neural network with five hidden units scored approximately 64% correct [Murphy and Aha, 1994]. This set has also been used for regression testing without the three-value aggregation; see Appendix D for the tests involving SECRET and this modeler's trees.

When the selector was set to regard all other attributes as inputs, and the regression tree set to prune only based on accuracy, both the unscaled and scaled trees performed on par – 64% accuracy, using 37 and 47 nodes respectively. Using the balanced pruner, both the unscaled and scaled trees yielded single-node trees, with the former scoring 55.47% and the latter 60.57%.

F.2 Liver

With the `liver` set and the binary `selector` attribute, classification performance was measured by treating the binary `selector` attribute as an integer attribute. This resulted in automatic rounding of the regression output to the nearest integer.

Treating all other attributes as inputs and using the balanced pruner, 71.3% accuracy was achieved without scaling; 68.7% with. Both cases involved single-node trees. Using the accuracy-only pruner, performance increased to approximately 74% in both cases with a five-node tree in the unscaled case and a nine-node tree in the scaled case.

The documentation does not specify any previous classification accuracy [Murphy and Aha, 1994]; a constant majority-class predictor would achieve a classification accuracy of 57.97%.

F.3 Page-blocks

On the `page-blocks` set, the class attribute was originally encoded as an integer from 1 to 5 corresponding to the unordered set of classes `text`, `horizontal line`, `graphic`, `vertical line`, and `picture`.

Therefore, the `class` attribute was split into five binary integer attributes corresponding to the five different classes. The models were collectively considered to have classified correctly if and only if the correct model for the corresponding class returned 1 and the other four models 0; a less strict approach would have been to use continuous outputs and select the class according to the maximal output while marking ties as incorrect.

The distribution among the classes is extremely skewed, with fully 4,913 of the 5,473 tuples labeled as `text` and only 28 of the tuples labeled as `graphic`. Hence, a constant majority-class predictor would achieve 76.6% accuracy.

Setting all other attributes as inputs, the system achieved a 94% classification accuracy without nonlinear scaling with a total of 35 nodes, but only 51.00% accuracy from 33 nodes with using the balanced pruner – a bizarre result given the skewed distribution of the classes.

With the accuracy-only pruner, both cases yielded a classification accuracy of 94%, but the axis scaling allows a more efficient solution with a total of 43 nodes instead of the 57 nodes in the unscaled scale.

Bibliography

BusinessWeek Online, April 2003. `http://www.businessweek.com/investor/content/apr2003/pi2003047_3105_pi010.htm`. 5.3.5

R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, June 1998. 3.1

H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, AC-19:716–723, 1974. 3.3

N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proc. of 28th ACM Symposium on the Theory of Computing*, pages 20–29, May 1996. C.2, C.2

S. Babu, M. Garofalakis, R. Rastogi, and A. Silberschatz. Model-based semantic compression for network-data tables. In *Proc. of NRDM 2001*, May 2001. 3.4, 4.1, 4.3.2, 4.4

B. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2:53–58, 1989. 3.1

B. T. Bartell, G. W. Cottrell, and R. K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *Proc. of ACM SIGIR*, 1992. 3.1

P. S. Bradley, U. M. Fayyad, and C. A. Reina. Scaling EM (Expectation-Maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, 1999. 4.1.1, 4.1.1, 4, 4.5.1, 4.5.3, B, D.1

L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone. *CART: Classification and Regression Trees*. Chapman & Hall / CRC Press, 1984. 3.2, 3.4

R. Calvo, M. Partridge, and M. Jabri. A comparative study of principal component analysis techniques. In *Proc. of Ninth Australian Conf. on Neural Networks*, February 1998. 3.1

D. Careggio. DJ 1985-2003. `http://lib.stat.cmu.edu`. 5.3.5

Central Intelligence Agency. The world factbook, 2001. `http://www.cia.gov/cia/publications/factbook/index.html`. 5.3.4

K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proc. of 26th International Conference on Very Large Data Bases*, pages 89–100. Morgan Kaufmann, September 2000. ISBN 1-55860-715-3. 3.1

K. Chang and J. Ghosh. Principal curves for nonlinear feature extraction and classification. *SPIE Applications of Artificial Neural Networks in Image Processing III*, 3307:120–129, 1998. 3.1

J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Proc. of the Sixth International Conference on Information and Knowledge Management*, pages 325–331, 1997. 3.4

S. Choenni. Design and implementation of a genetic-based algorithm for data mining. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proc. of 26th International Conference on Very Large Data Bases*, pages 33–42. Morgan Kaufmann, September 2000. ISBN 1-55860-715-3. 3.2

A. C. Cohen Jr. Estimating the mean and variance of normal populations from singly truncated and doubly truncated samples. *Annals of Mathematical Statistics*, 21 (4):557–569, December 1950. 4, A, A, A

S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss lemma. Technical Report TR-99-006, UC Berkeley, 1999. 3.1

D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 580–587. Morgan Kaufmann, San Mateo, CA, 1993. 3.1

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977. 4.1.1, 4, B

A. Dobra and J. Gehrke. SECRET: a scalable linear regression tree algorithm. In *Proc. of ACM SIGKDD*, pages 481–487, July 2002. 3.2, 4.3, 4.5.1, D, D.4

F. Esposito, D. Malerba, and G. Semeraro. Multistrategy learning for document recognition. *Applied Artificial Intelligence*, 8:33–84, 1994. 5.3.8

C. Faloutsos and K.-I. D. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM SIGMOD*, pages 163–174, May 23-25 1995. 3.1

O. W. Gilley and R. K. Pace. On the harrison and rubinfeld data. *Journal of Environmental Economics of Management*, 31:403–405, 1996. 5.3.6

I. Greenwald. Sears, Chrysler lead blue-collar rally, April 2002. `http://www.smartmoney.com/bn/index.cfm?story=20020410085106`. 5.3.5

M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 58–66, May 2001. D.2

R. Hackley. GE wins Honeywell's hand, Welch stays on. `http://www.smartmoney.com/bn/index.cfm?story=20001019050654`. 5.3.5

M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454), 2001. 3.3

D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics of Management*, 5:81–102, 1978. 5.3.6

J. Haslett and A. E. Raftery. Space-time modeling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics*, 38:1–50, 1989. 5.3.9

C. hung Cheng, A. W. chee Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. of 5th International Conference on Knowledge Discovery and Data Mining*, pages 84–93, August 1999. 3.1

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999. 3.1

A. Hyvärinen, J. Karunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001. 3.1

H. V. Jagadish, J. Madar, and R. Ng. Semantic compression and pattern extraction with fascicles. In *Proc. of the 25th International Conference on Very Large Data Bases*, pages 186–197. Morgan Kaufmann, September 1999. ISBN 1-55860-615-7. 3.4, 3.5

H. V. Jagadish, R. T. Ng, B. C. Ooi, and A. K. H. Tung. ItCompress: an iterative semantic compression algorithm. In *Proc. of International Conference on Data Engineering*, March 2004. 3.5

T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Proc. of 16th Conference on Uncertainty in Artificial Intelligence*, July 2000. 3.1

N. Johnson and S. Kotz. *Continuous univariate distributions*. Houghton Mifflin, 1970. 4.1.1, 4.1.2, B

T. Johnson. Honeywell, United Tech in talks, October 2000. `http://money.cnn.com/2000/10/19/deals/honeywell`. 5.3.5

I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986. 3.1

M. Jones. Using recurrent networks for dimensionality reduction. AI Tech Report 1396, MIT, 1992. 3.1

Z. A. Karian, E. J. Dudewicz, and P. McDonald. The extended generalized lambda distribution system for fitting distributions to data: history, completion of theory, tables, applications, the "final word" on moment fits. *Communications in Statistics: Simulation and Computation*, 25(3):611–642, 1996. 3

E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data (SIGMOD '01)*, 2001. 3.1

E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proc. of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, August 2004. 3.3

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997. 3.1

M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers J.*, 37(2):233–243, 1991. 3.1

J. F. Kreider and J. S. Haberl. The great energy predictor shootout – the first building data analysis and prediction competition, June 1993. ASHRAE meeting. 5.3.3

J. B. Kruskal. Non metric multidimensional scaling: a numerical method. *Psychometrika*, 29:115–129, 1964. 3.1

K. Lang. NewsWeeder: Learning to filter netnews. In *Proc. of 12th International Conference on Machine Learning*, pages 331–339, 1995. 3.1

W. Loh and Y. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997. D.1

P. Murphy and D. Aha. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1994. 1, 5.3.1, 5.3.6, 5.3.7, 5.3.8, D.4, F.1, F.2

W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn, and W. B. Ford. The population biology of abalone (*haliotis* species) in Tasmania. I. Blacklip Abalone (*h. rubra*) from the north coast and islands of Bass Strait. Technical Report 48, Sea Fisheries Division, 1994. ISSN 1034-3288. 5.3.1

J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965. 4.1.3

M. Partridge and R. Calvo. Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis*, 2(3), 1998. 3.1

L. Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, 1994. 5.3.3

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993. 3.2, D.2

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978. 3.3

V. Roth. The generalized LASSO: a wrapper approach to gene selection for microarray data. Technical Report IAI-TR-2002-8, University of Bonn, 2002. 3.1

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, December 2000. 3.1

J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18:401–409, 1969. 3.1

M. Sato. Fast learning of on-line EM algorithm. Technical Report TR-H-281, ATR Human Information Processing Research Laboratories, 1999. 4

Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. 3.1

M. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise.* W.H. Freeman and Company, New York, 1991. C.1

T. Takahashi and R. Tokunaga. Nonlinear dimensionality reduction by multi layer perceptron using superposed energy. In *Proc of. International Symposium on Nonlinear Theory and its Applications*, volume 2, pages 863–866, 1999. 3.1

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. 290:2319–2322, December 2000. 3.1

B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45(3):279–299, December 2001. 4

L. F. R. A. Torgo. *Inductive Learning of Tree-based Regression Models*. PhD thesis, University of Porto, September 1999. 3.2, 4.3

C. Traina Jr., A. J. M. Traina, L. Wu, and C. Faloutsos. Fast feature selection using the fractal dimension. In *Proc. of the XV Brazilian Symposium on Databases*, pages 158–171, October 2000. 3.1, 4.2.2, C.1, C.1

I. Tsamardinos and C. F. Aliferis. Towards principled feature selection: Relevancy. In *Proc. of Artificial Intelligence and Statistics 2003*, January 2003. 3.1

C. S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *Computer Journal*, 42(4):270–283, 1999. 3.3

J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proc. of 5th International Conference on Knowledge Discovery and Data Mining*, pages 307–311, August 1999. 3.1

S. M. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, December 1995. 3.2

D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, January 1997. 4.2.2

A. Wong, L. Wu, and P. B. Gibbons. Fast estimation of fractal dimension and correlation integral on stream data. In *Proceedings of the Second Workshop on Fractals, Power Laws and Other Next Generation Data Mining Tools*, August 2003. C.2, C.2

A. Wong, L. Wu, and P. B. Gibbons. Fast estimation of fractal dimension and correlation integral on stream data. *Information Processing Letters*, 93:91–97, 2005. C.2, C.2

L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996. 4.1.1