4. Comparing Methods

We have described eight methods so far: Loess, Additive Models (AM), PPR, Neural Nets (NN), ACE, AVAS, Recursive Partitioning Regression (RPR), and MARS. In addition, a practitioner might consider traditional stepwise linear regression (SLR) and multiple linear regression (MLR).

One would like to make comparisons among these. In general, one would like to have a practicum for making comparisons among complex statistical procedures that are too difficult to analyze using theory.

This section describes a model simulation experiment, and it lays out the general strategies for such comparisons.

Banks, Olszewski, and Maxion (2003; *Communications in Statistics: Simulation and Computation*, **32**, 541-571) describe a $10 \times 5 \times 3^4$ designed experiment to compare the methods.

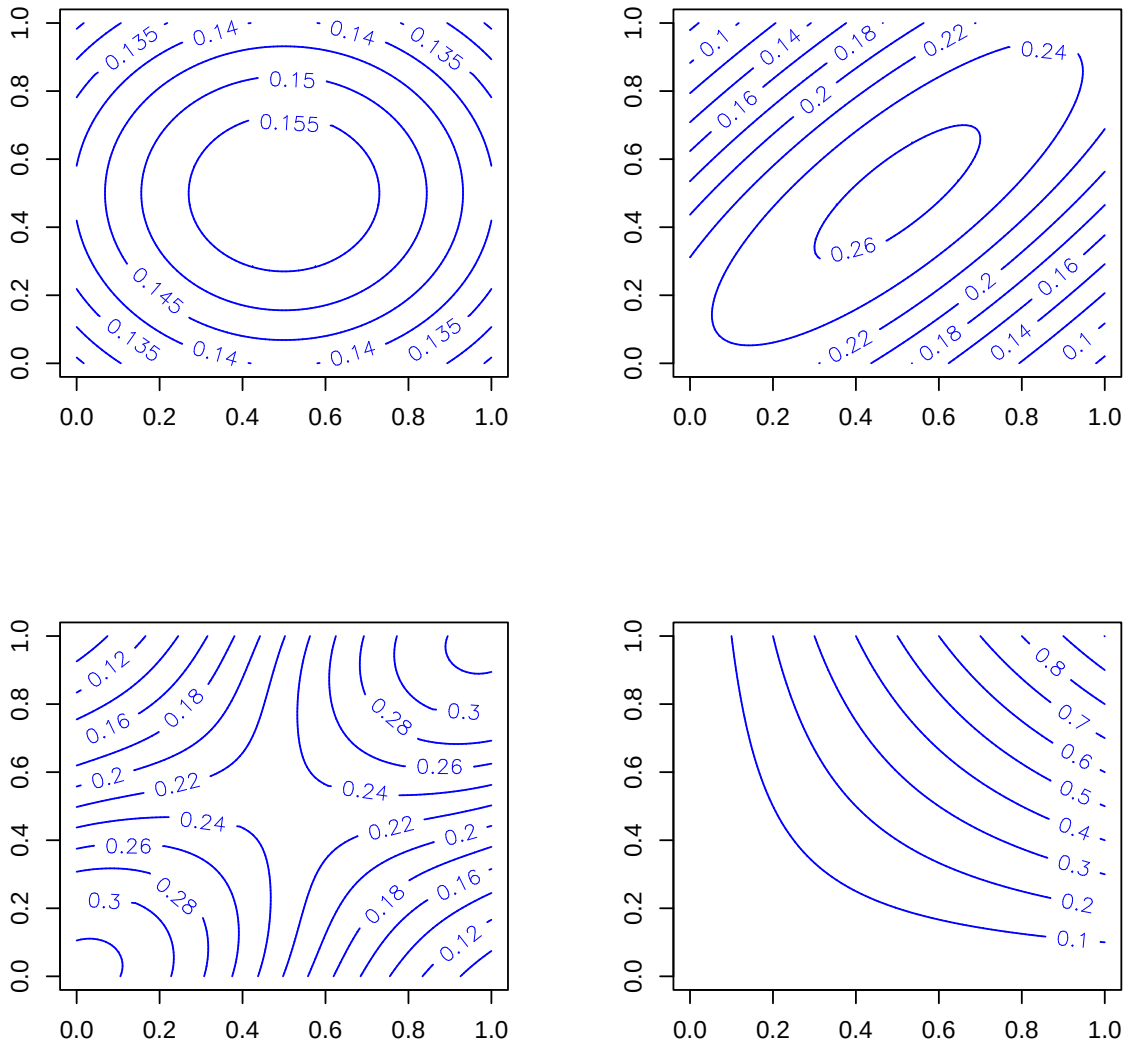
The six factors in the experiment were:

1. The ten regression methods.
2. Five target functions.
3. The dimension: $p = 2, 6, 12$.
4. The sample size: $n = 2^p k$ for $k = 4, 10, 25$.
5. The proportion of spurious variables: this took the values all, half, and none.
6. The noise: this is the variance in ϵ, the additive Gaussian noise, and took the values $\sigma = .02, .1, .5$.

The target functions consist multivariate functions that approximate features likely to be of scientific interest. The five functions are

- the constant function,
- the hyperflat,
- the standard normal recentered to $(.5, .5)'$,
- a normal centered $(.5, .5)'$ with covariance matrix $.8\boldsymbol{I}$
- a mixture of two standard normals, one centered at $(0, 0)'$ and the other at $(1, 1)'$
- the product function $x_1 x_2 \cdots x_p$.

The constant function is especially important because in that case the explanatory values are irrelevant, but some methods tend to discover spurious signal in the noise.


0.0
0.2
0.4
0.6
0.8
1.0
0.135
0.14
0.15
0.155
0.145
0.1
0.16
0.18
0.22
0.24
0.26
0.12
0.28
0.3
0.7
0.5

The simulation experiment had the following steps:

1. Generate a random sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ uniformly in the unit cube in $\mathbb{R}^p$.
2. Generate random $N(0, \sigma^2)$ errors.
3. Calculate $Y_i = f(\boldsymbol{x}_i) + \epsilon_i$, where f is one of the target functions.
4. Apply each of the regression methods to obtain estimate $\hat{f}$ of f.
5. Estimate the integrated squared error of each $\hat{f}$ over the unit cube (Monte Carlo, using 10,000 random points).
6. Repeat steps 1-5 twenty times and average to obtain an estimate of MISE.

Note that this procedure reuses the same data points across the estimators, which reduces the variance in contrasts.

σ	n	Sp.	Meth.	MISE $p=2$	SE	MISE $p=6$	SE	MISE $p=12$	SE
M	S	A	MLR	(27.95)	7.02	(3.00)	.26	(.09)	.01
M	S	A	SLR	(32.14)	10.65	(3.00)	.26	(.09)	.01
M	S	A	ACE	53.12	7.36	15.06	1.14	.41	.02
M	S	A	AM	(27.91)	7.01	(3.00)	.26	(.09)	.01
M	S	A	MARS	158.23	35.32	18.36	1.88	.30	.02
M	S	A	RPR	1248.37	307.78	176.32	10.89	61.53	.78
M	S	A	PPR	55.08	13.14	19.24	8.68	.11	.01
M	S	A	LOESS	59.50	9.12	9.14	.52	*	*
M	S	A	AVAS	79.40	18.14	14.73	.95	.38	.02
M	S	A	NN	124.03	13.05	100.42	2.43	*	*

σ	n	Sp.	Meth.	MISE $p=2$	SE	MISE $p=6$	SE	MISE $p=12$	SE
M	S	H	MLR	27.95	7.02	3.00	.26	(.09)	.01
M	S	H	SLR	(23.23)	6.75	(2.39)	.30	(.08)	.01
M	S	H	ACE	42.42	5.92	14.58	1.12	.40	.02
M	S	H	AM	27.68	7.00	3.00	.26	(.09)	.01
M	S	H	MARS	(17.99)	3.24	11.00	1.68	.40	.02
M	S	H	RPR	1821.76	64.32	239.41	4.50	100.04	2.47
M	S	H	PPR	35.31	5.61	13.95	8.09	.11	.01
M	S	H	LOESS	59.50	9.12	9.14	.52	*	*
M	S	H	AVAS	74.93	14.13	15.02	1.08	.36	.01
M	S	H	NN	103.08	8.41	94.92	2.65	*	*

MLR, SLR, and AM perform similarly over all situations considered, and represent broadly safe choices. They are never disastrous, though rarely the best (except for MLR when the target function is linear and all variables are used). For the constant function, SLR shows less overfit than MLR, which is better than AM; however, it is easy to find functions for which AM would outperform both MLR and SLR. SLR is usually slightly better with spurious variables, but its strategy becomes notably less effective as the number of spurious variables increases, especially for non-linear functions. All three methods have greatest relative difficulty with the product function, which has substantial curvature.

On theoretical grounds ACE and AVAS should be similar, but this is not always borne out. ACE is decisively better for the product function, and AVAS for the constant function. ACE and AVAS are the best methods for the product function (as expected—the log transformation produces a linear relationship), but among the worst for the constant function and the mixture of Gaussians; in other cases, their performance is not remarkable. Both methods are fairly robust to spurious variables.

Contrary to expectation, MARS does not show well in higher dimensions, especially when all variables are used, and especially for the linear function. However, for lower dimensions, MARS shows adequate performance across the different functions. MARS is well-calibrated for the constant function when $p = 2$, but finds spurious structure for larger values, which may account for some of its failures.

RPR was consistently bad in low dimensions, but sometimes stunningly successful in high dimensions, especially when all variables were used. Surprisingly, its variable selection capability was not very successful (MARS's implementation clearly outperforms it). Perhaps the CART program, with its flexible pruning, would surpass RPR, but previous experience with CART makes us dubious. Unsurprisingly, RPR's design made it uncompetitive on the linear function.

PPR and NN are theoretically similar methods, but PPR was clearly superior in all cases except the correlated Gaussian. This may reflect peculiarities of the Cascor implementation of neural nets. PPR was often among the best when the target function was the Gaussian, correlated Gaussian, or mixture of Gaussians, but among the worst with the product and constant functions. PPR's variable selection was generally good. In contrast, NN was generally poor, except for the correlated Gaussian when $p = 2, 6$ and all variables are used and when $p = 6$ and half the variables are used. The correlated Gaussian lends itself to approximation by a small number of sigmoidal functions whose orientations are determined by the data.

LOESS does well in low dimensions with the Gaussian, correlated Gaussian, and mixture of Gaussians. It is not as successful with the other target functions, especially the constant function. Often, it is not bad in higher dimensions, though its relative performance tends to deteriorate.

For the constant function, MARS is good when $p = 2$, SLR is good when $p = 6$, and RPR is good when $p = 12$. For the linear function, MLR and SLR are consistently strong. For the Gaussian function, with all variables used, LOESS and MARS are good when $p = 2$, SLR is good when $p = 6$, and RPR is good when $p = 12$; when half of the variables are used, MARS and PPR perform well. For the correlated Gaussian, with all variables used, LOESS works well for $p = 2$, LOESS and NN for $p = 6$, and ACE or AVAS for $p = 12$; with half the variables used, MARS is reliably good. For the mixture of Gaussians, with all variables used, LOESS works well for $p \leq 6$, and RPR for $p = 12$; with half of the variables, MARS is consistently good. There is considerable variability for the product function, but ACE is broadly superior.

Two kinds of variable selection strategies were used by the methods: global variable selection, as practiced by SLR, ACE, AVAS, and PPR, and local variable reduction, as practiced by MARS and RPR. Generally, the latter does best in high dimensions, but performance depends on the target function.

LOESS, NN, and sometimes AVAS proved infeasible in high dimensions. The number of local minimizations in LOESS grew exponentially with p. Cascor's demands were high because of the cross-validated selection of the hidden nodes; alternative NN methods fix these a priori, making fewer computational demands, but this is equivalent to imposing strong, though complex, modeling assumptions. Typically, fitting a single high-dimensional dataset with either LOESS or NN took more than two hours. AVAS was faster, but the combination of high dimension and large sample size also required substantial time.

5. Local Dimension

Nearly all methods for multivariate nonparametric regression do local model fitting (otherwise, they must make strong model assumptions, such as multiple linear regression). Local fitting is most likely to succeed if the response function $f(\boldsymbol{x})$ has **locally-low dimension**.

A function $f : \mathbb{R}^p \to \mathbb{R}$ has locally-low dimension if there exists a set of regions $R_1, R_2, \ldots$ and a set of functions $g_1, g_2, \ldots$ such that $\bigcup R_i \approx \mathbb{R}^p$ and for $\boldsymbol{x} \in R_i$, $f(\boldsymbol{x}) \doteq g_i(\boldsymbol{x})$ where g_i depends upon only q components of $\boldsymbol{x}$ for $q \ll p$.

This uses a vague sense of approximation, but it can be made precise.

The following functions have high local dimension:

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j \text{ for } \beta_j \neq 0$$

$$f(\boldsymbol{x}) = \prod_{j=1}^{p} x_j.$$

In contrast, the following functions have locally-low dimension:

$$f(\boldsymbol{x}) = \begin{cases} 3x_1 & \text{if } x_1 + x_2 < 7 \\ x_2^2 & \text{if } x_1 + x_2 > 7 \\ x_1 + & \text{if } x_1 = x_2 \end{cases}$$

$$f(\boldsymbol{x}) = \sum_{k=1}^{m} \alpha_k I_{R_k}(\boldsymbol{x}).$$

Regression analysis in high dimensions seems impossible without strong model assumptions or locally-low dimension.

Before attempting a statistical analysis, it would be good to know whether the local dimension is low. (If not, one should anticipate disappointment.) So how can one estimate the local dimension?

Let $\{(y_i, \boldsymbol{x}_i)\}$ denote the sample. Then iterate the following steps M times.

1. Select a random point $\boldsymbol{X}_m^*$ in the convex hull of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, for $m = 1, \ldots, M$
2. Find a ball centered at $\boldsymbol{X}^*$ that contains exactly k points (say $k = 4p$).
3. Perform a principal components regression on the k points within the ball.
4. Let c_m be the number of principal components needed to explain a fixed percentage (say 80%) of the variance in the Y_i values.

This suggests that the average of $c_1, \ldots, c_M$ may be a useful estimate of the average local dimension of f.

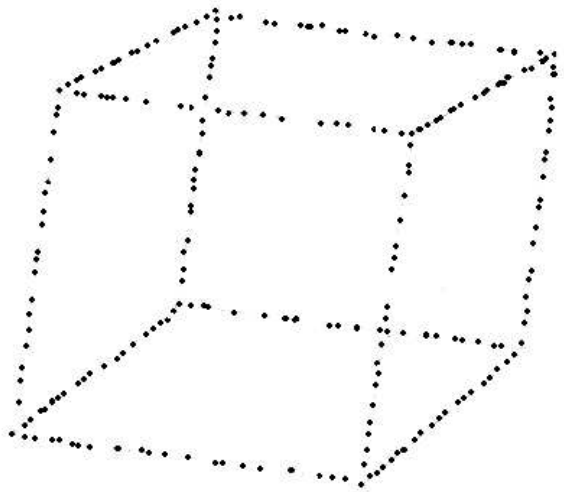
This heuristic approach assumes a locally linear functional relationship for points within the ball. The Taylor series motivates this, but the method will break down for some pathological functions or if the data are too sparse.

To test the approach, Banks and Olszewski (2003; *Statistical Data Mining and Knowledge Discovery*, 529-548, Chapman & Hall) performed a simulation experiment in which random samples were generated from q-cube submanifolds in $\mathbb{R}^p$, and the approach described above was used to estimate q.

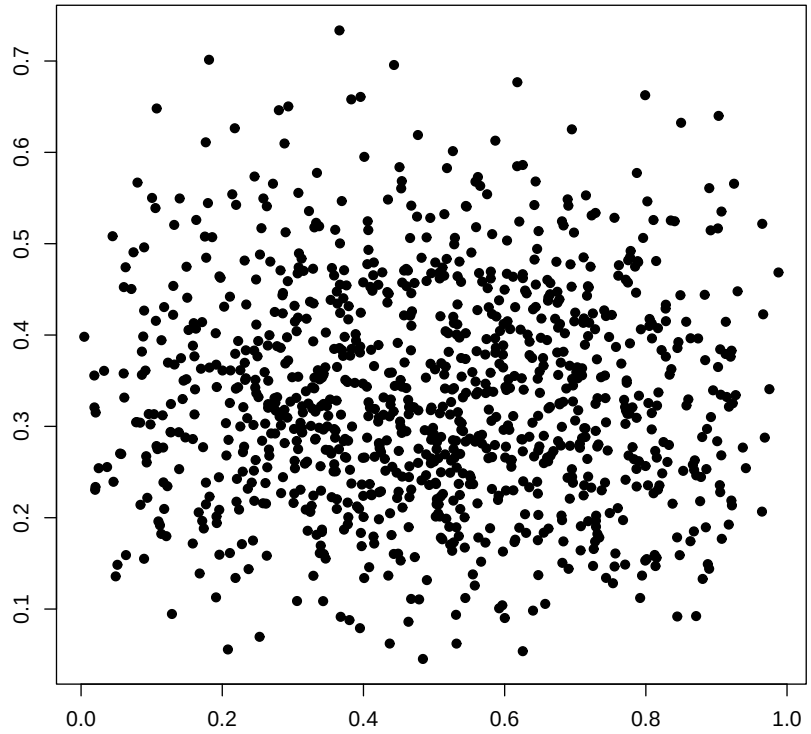
A **q-cube** in $\mathbb{R}^p$ is the q-dimensional boundary of a p-dimensional cube. Thus:

- a 1-cube in $\mathbb{R}^2$ is the perimeter of a square,
- a 2-cube in $\mathbb{R}^3$ are the faces of a cube,
- a 3-cube in $\mathbb{R}^3$ is the entire cube.

The following figure shows a 1-cube in $\mathbb{R}^3$, tilted x degrees from the natural axes in each coordinate.



The following figure shows a 1-cube in $\mathbb{R}^{10}$, tilted x degrees from the natural axes in each coordinate.



Diaconis and Freedman (1984; *Annals of Statistics*, **12**, 793-815) show that in high-dimensions, nearly all projections look normal.

The simulation study generated $10 * 2^q$ points at random on each of the $2^{p-q} \begin{pmatrix} p \\ q \end{pmatrix}$ sides of a q-cube in $\mathbb{R}^p$. Then iid $N(\mathbf{0}, .25\boldsymbol{I})$ noise was added to each observation and the principal components approach was used to estimate q for all values of q between 1 and p for $p = 1, \ldots, 7$.

The first following table shows that the method was reasonably successful in estimating the local dimension. The estimates are biased, since the principal components analysis identified the number of linear combinations needed to explain only 80% of the variance. One should probably do some kind of bias correction to account for the underestimate.

The second following table estimates the proportion of the data region that is sparse; the method puts random balls of fixed size into the dataset and counts the number of times the ball contains fewer than $2p$ observations.

q							
7							5.03
6						4.25	4.23
5					3.49	3.55	3.69
4				2.75	2.90	3.05	3.18
3			2.04	2.24	2.37	2.50	2.58
2		1.43	1.58	1.71	1.80	1.83	1.87
1	.80	.88	.92	.96	.95	.95	.98
	p=1	2	3	4	5	6	7

The value of p indicates the apparent dimension, while q is the true dimension of the data. Each entry is an estimate of q, and the largest standard error in the table is .03.

q							
7							41.0
6						39.1	52.2
5					34.4	45.3	53.2
4				32.3	36.2	46.1	55.9
3			29.1	27.0	34.7	48.6	57.9
2		28.4	26.5	32.0	41.6	46.6	55.2
1	28.5	40.1	45.8	51.0	51.3	51.0	52.5
	p=1	2	3	4	5	6	7

The value of p indicates the apparent dimension, while q is the true dimension of the data. The number in a row indicates the proportion of spheres that did not contain at least $2p$ observations.