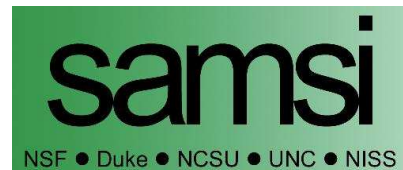


# *Cherry Picking: A New Robustness Tool*

David Banks and Leanna House

*Institute of Statistics & Decision Sciences*  
**Duke University**



# 1. Background

This problem arose during the SAMSI year-long program in data mining.

Suppose one has a large, complex dataset that contains multiple kinds of structures and possibly noise, e.g.:

- 40% of the data follow

$$Y = \alpha_0 + \sum_{i=1}^p \alpha_i X_i + \epsilon$$

- 30% of the data follow

$$Y = \beta_0 + \sum_{i=1}^p \beta_i X_i + \epsilon$$

- 30% of the data are noise.

What can one do to analyze cases like this? Assume the analyst has little prior knowledge of the kinds of structure that might be present.

The standard approach in linear regression is to use S-estimators, which look for the thinnest strip (think of a transparent ruler) which covers some prespecified (but larger than 50%) fraction of the data.

This strategy breaks down in high dimensions, or when the structures of interest contain less than 50% of the sample, or when fitting complex nonlinear models.

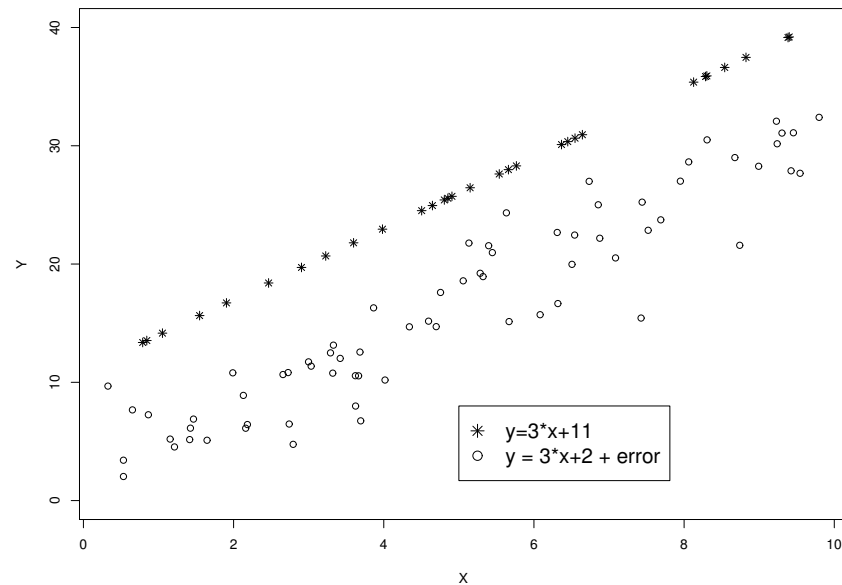
We present a solution strategy that applies to more general cases, including:

- Linear and non-linear regression in high dimensions
- Multidimensional scaling
- Cluster analysis

We believe the approach can be extended to other cases as well.

## 2. Regression

Consider the graph below, for a simple regression problem. It is clear that the data come from two different models, and that any naive attempt at regression will miss both of the interesting structures and find some kind of average solution.



In the linear regression scenario, assume we have  $n$  observations:  $\{Y_i, \mathbf{X}_i\}$  and that  $Q$  percent of these follow the model

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i \quad \text{where} \quad \epsilon_i \sim N(0, \sigma)$$

where  $Q$ ,  $\beta$ , and  $\sigma$  are unknown.

We say that  $Q\%$  of the data are “good” and the rest are “bad”.

### Simple Idea:

Start small, with a subsample of only **good** observations

⇒ add only **good** observations

⇒ end with a large subsample of **good** observations.

General procedure:

- 1.** Strategically choose an initial set of  $d$  starting subsamples  $S_j$ , each of size  $m$
- 2.** Grow the subsamples by adding consistent data
- 3.** Select the largest subsample.

## 2.1 Algorithmic Details

One starts with a guess about  $Q$ , the fraction of good data. In practice, this is unknown, so one might pick a value that is reasonable given

- domain knowledge about the data collection
- scientific interest in a fraction of the data.

From the full dataset  $\{Y_i, \mathbf{X}_i\}$  one selects, without replacement,  $d$  subsamples  $S_j$  of size  $m$ .

One needs to choose  $d$  and  $m$  to ensure that at least one of the starting subsamples  $S_j$  has a very high probability  $C$  of consisting entirely of good data (i.e., data that come from the model).

Preset a probability  $C$  that determines the chance that the algorithm will work.

The value  $m$ , which is the size of the starting-point random subsamples, should be the smallest possible value that allows one to calculate a goodness-of-fit measure. In the case of multiple linear regression, that value is  $p + 2$ , and the natural goodness-of-fit measure is  $R^2$ .

One solves the following equation for  $d$ :

$$C = \mathbb{P}[\text{at least one of } S_1, \dots, S_d \text{ is all good}] = 1 - (1 - Q^{p+2})^d$$

.

Example:  $Q = .8$ ,  $c = .95$ ,  $m = 3$  ( $p = 1$ ):

$$.95 = 1 - [1 - (.8)^{p+2}]^d \quad \rightarrow \quad d = 5$$

Given the  $d$  starting-point subsamples  $S_j$ , one grows each one of them by adding observations that do not lower the goodness-of-fit statistic ( $R^2$ ).

Conceptually, for a particular  $S_j$ , one could cycle through all of the observations, and on each cycle augment  $S_j$  by adding the observation that provided the largest value of  $R^2$ . This cycling would continue until no observation can be added to  $S_j$  without decreasing the  $R^2$ .

One does this for all of the subsamples  $S_j$ . At the end of the process, each augmented  $S_j$  would have size  $m_j$  and goodness-of-fit  $R_j^2$ . The augmented subsample that achieves a large value of  $m_j$  and a large value of  $R_j^2$  is the one that captures the most important structure in the data.

Then one can remove the data in  $S_j$  and iterate to find the next-most important structure in the dataset.

In practice, the conceptual algorithm which adds one observation per cycle is expensive when the dataset is large or when one is fitting a complex model (e.g., doing MARS fits rather than multiple regression). For this reason, we use a two-step procedure to add observations.

### Fast Search

- Sequentially sweep through all observations not in  $\xi_i$
- If the observation [improves the FM] or [only lowers FM by a small amount  $\eta$ ],  
→ add observation to  $\xi_j$   
→  $m_j = m_j + 1$

If  $m_j < kn$  then implement slow search

### Slow Search

- Add the observation that improves the FM the most or decreases the FM the least (regardless of  $\eta$ )
- Repeat until  $m_j = kn$

The analyst can pick a value for  $\eta$  that seems appropriately small, and a value for  $k$  that seems appropriately large. These choices determine the runtime of the algorithm and should reflect practical constraints.

The fast search is greedy, and the order of observations in the cycling matters. The slow search is less greedy; order does not matter, but it adds myopically. The fast search can add many observations per cycle through the data, but the slow search always adds exactly one.

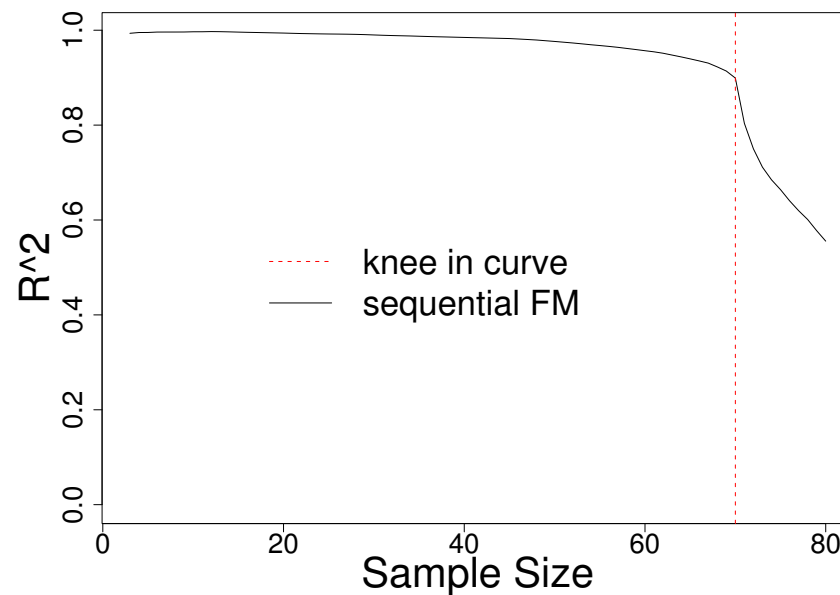
If speed is truly important, then there are other ways to accelerate the algorithm. A standard strategy would be to increase the number of starting-point subsamples and combine those that provide similar models and fits as they grow.

The main concern is not to enumerate all  $\binom{n}{Qn/100}$  possible subsamples.

Some further comments are useful.

- 1.** One does not need to terminate the search at some preset value  $kn$ ; one can just grow until the goodness-of-fit measure deteriorates too much.
- 2.** The goodness-of-fit measure should not depend upon the sample size. For  $R^2$  this is easy, since it is just the proportion of variation in  $Y$  explained by  $X$ . For larger  $p$ , if one is doing stepwise regression to select variables, then one wants to use an AIC or Mallows'  $C_p$  statistic to adjust the tradeoff in fit between the number of variables and the sample size.
- 3.** Other measures of fit are appropriate for nonparametric regression, such as cross-validated within-subsample squared error. But this adds to the computational burden.
- 4.** One can and should monitor the fit as new observations are added. When one starts to add bad data, this is quickly visible, and there is a clear “slippery-slope” effect.

To see how the slippery-slope occurs, and the value of monitoring fit as a function of order of selection, consider the plot below. This plot is based on using  $R^2$  for fitness with the double-line data shown previously. The total sample size is 80, and 70 observations were generated exactly on a line, as indicated by the knee in the curve.



### 3. Application: Multidimensional Scaling

Multidimensional scaling (MDS) starts with a proximity matrix that gives “distances” between all pairs in a set of objects. These distances need not satisfy a true metric, but the analyst generally believes that they are close to a metric.

The purpose of MDS is to find a low-dimensional plot of the objects such that the inter-object distances are as close as possible to the values given in the proximity matrix. That representation automatically puts similar objects near each other.

MDS seeks to find pseudo-objects in a low-dimensional space whose distances are close, in terms of least squares, to the values in the proximity matrix. This is done by minimizing the stress function:

$$\text{Stress}(z_1, \dots, z_n) = \left[ \sum_{i \neq i'} (d_{ii'} - \|z_i - z_{i'}\|) \right]^{1/2}$$

where  $z_i$  is the location assigned to pseudo-object  $i$  in the low-dimensional space and  $d_{ii'}$  is the entry in proximity matrix.

The classic example is to take the entries in the proximity matrix to be the drive-time between pairs of cities. This is not a perfect metric, since roads curve, but it is approximately correct. MDS finds a plot in which the relative position of the cities looks like it would on a map (except the map can be in any orientation; north and south are not relevant).

MDS seeks is extremely susceptible to bad data. For example, if one had a flat tire while driving from Rockville to DC, this would create a seemingly large distance. The MDS algorithm would distort the entire map in an effort to put Rockville far from DC and still respect other inter-city drive times.

A very small proportion of outliers, or objects that do not fit well in a low-dimensional representation, can completely wreck the interpretability of an MDS plot. In many applications, such as text retrieval, this is a serious problem.

## 3.1 MDS Example

To extend cherry-picking approach to MDS, we took an example dataset consisting of the latitudes and longitudes of 99 eastern United States cities. The Euclidean distances between these cities generated the proximity matrix; the only stress in the MDS map is due to the curvature of the earth.

We then perturbed the proximity matrix by inflating, at random, a proportion  $1 - Q$  of the entries:

True $1-Q$ (%)	Distance Distortion (%)	Original Stress
2	150	1.028
	500	2.394
10	150	1.791
	500	28.196
30	150	3.345
	500	9.351

To make things more interesting, we use not the traditional MDS using the stress measure defined previously, but rather Kruskal-Shephard non-metric scaling, in which one finds  $\{z_i\}$  to minimize

$$\text{Stress}_{KS}(z_1, \dots, z_n) = \frac{\sum_{i \neq i'} [\theta(\|z_i - z_{i'}\|) - d_{ii'}]^2}{\sum_{i \neq i'} d_{ii'}^2}$$

where  $\theta(\cdot)$  is an arbitrary increasing function fit during the minimization. The result is invariant to monotonic transformations of the data, which is why it is nonparametric.

This minimization uses an alternating algorithm that first fixes  $\theta(\cdot)$  and finds the  $\{z_i\}$ , and then fixes the  $\{z_i\}$  and uses isotonic regression to find  $\theta(\cdot)$ .) This shows that the algorithm can be used in complex fits.

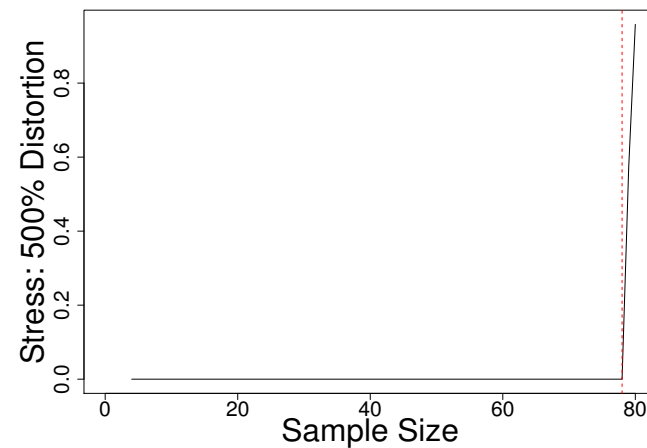
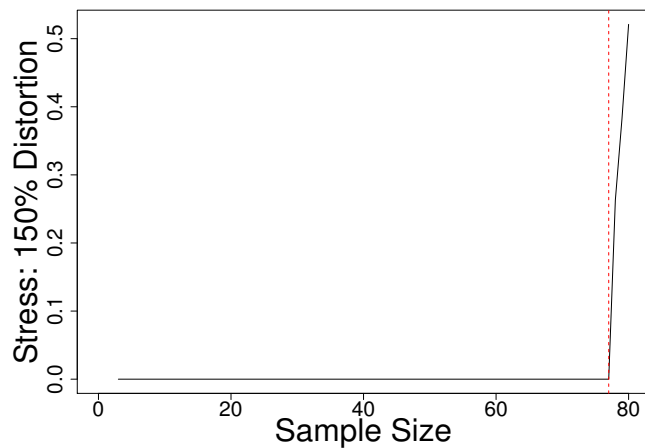
Our goal is to cherry-pick the largest subset of cities whose intercity distances can be represented with little stress.

In MDS, the size  $m$  of the initial subsamples is 4 (since three points are always coplanar). We took  $C = .99$  as the prespecified chance of getting at least one good subsample, and the table below shows the results.

True 1- $Q$ (%)	Distance Distortion (%)	Original Stress	$n^a$	$n^*$	Final Stress
2	150	1.028	80	80	4.78e-12
	500	2.394	80	80	4.84e-12
10	150	1.791	80	80	4.86e-12
	500	28.196	80	80	4.81e-12
30	150	3.345	80	77	4.86e-12
	500	9.351	80	78	4.78e-12

- Note: The stress of the undistorted dataset was  $8.42 \times 10^{-12}$ .

As before, one should inspect order-of-entry plots that display the stress against the cities chosen for inclusion. The following two plots are typical, and show the knee in the curve that occurs when one begins to add bad cities.



## 4. Summary / Discussion

- We have described a simple strategy for identifying primary structure in complex datasets.
- The calculations are practical in computer-intensive applications, but one needs to use relatively greedy search algorithms to select observations for inclusion.
- The main computational problem is to scale the algorithm to accommodate very large samples, but there are obvious ways to address this.
- We can make probabilistic statements about the chance of having a good starting-point subsample, and this almost leads to a probabilistic guarantee on the result, but not quite.
- Simulation indicates this works well across a range of problems and situations.
- Once structure is discovered in the data, it can be removed and the process repeated to find second-order structure.
- The same approach can be used in cluster analysis, where fit is measured by the ratio of within-cluster variation to between-cluster variation.