Cherry-picking for complex data: robust structure discovery

BY DAVID L. BANKS1,*, LEANNA HOUSE2 AND KEVIN KILLOURHY3

1 Department of Statistical Science, Duke University, Durham, NC 27705, USA
2 Department of Statistics, Virginia Tech, Blacksburg, VA 24061, USA
3 Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Complex data often arise as a superposition of data generated from several simpler models. The traditional strategy for such cases is to use mixture modelling, but it can be problematic, especially in higher dimensions. This paper considers an alternative approach, emphasizing data exploration and robustness to model misspecification. The strategy is applied to problems in regression, cluster analysis and multidimensional scaling. The approach is illustrated through simulation and the analysis of several datasets.

Keywords: cluster analysis; elemental sets; mixture models; multidimensional scaling; regression

1. Introduction

A key theme that emerged during the Isaac Newton Institute’s programme on statistical theory and methods for complex, high-dimensional data was the importance of sparsity. There are several kinds of sparsity, including the following.

(a) Global sparsity

The curse of dimensionality (cf. Hastie et al. 2002) arises when doing highly multivariate inference without strong parametric assumptions. Global sparsity assumes that, although one has $p$-dimensional data, in fact the inference of interest depends only on relationships in a $q$-dimensional subspace, where $q \ll p$. The difficulty is to identify that low-dimensional space. Much of the recent research has focused on this problem: the least absolute shrinkage and selection operator (LASSO; Tibshirani 1996) and least angle regression software (Efron et al. 2004) address this in the context of multiple regression, where the response is a linear function of an unknown but small subset of a very large number of potential predictors. Much of the theory in this area is related to ‘sufficient dimension reduction’ (SDR); see Bondell & Li (2009). A motivating application is genomics; one observes expression levels for thousands of genes but only a handful are likely to be predictive of a specific phenotypic trait.

*Author for correspondence (banks@stat.duke.edu).

One contribution of 11 to a Theme Issue ‘Statistical challenges of high-dimensional data’.
(b) Local sparsity

In many situations one cannot expect global sparsity; all of the variables are important in some circumstances. In these cases one can hope for local sparsity, in which there is a partition of the \( p \)-dimensional space such that, within each region, the response depends upon only a small number of the explanatory variables. This perspective leads to classification and regression trees (Breiman et al. 1984) and random forests (Breiman 2001). For example, if the response variable is lung cancer, then one can partition the space by age and smoking status: for older smokers, the relevant predictor is the number of cigarettes used per day; for older non-smokers, the relevant predictor is exposure to second-hand smoke or workplace carcinogens; for young people, genetic factors are the most predictive covariates.

(c) Mixture sparsity

A third kind of sparsity occurs when the observed data arise from several simple models. Each model depends upon only a few of the covariates, but different sets of covariates may be involved for different models, and, even if two models employ the same covariates, the functional relationships are different. Such problems are traditionally addressed through mixture modelling (Titterington et al. 1985), using either frequentist (e.g. Lindsay 1995) or Bayesian (e.g. Richardson & Green 1997) methods. These problems can also be framed in terms of multivariate SDR.

This paper focuses upon mixture sparsity, and proposes an algorithmic alternative to standard mixture model analysis that is related to elemental set methods (cf. Theil 1950; Atkinson 1986; Hawkins 1993a). The approach generalizes to different kinds of statistical methods, and we illustrate these extensions on several different datasets. The analyses show that the approach has attractive robustness properties and supports exploratory data analysis.

Mixture sparsity is a natural assumption for many practical situations. For example, in survey data, it is usual that a significant proportion of the responses are so flawed as to be irrelevant or even misleading to the analysis. In such cases one has a mixture with at least two components; one component contains the signal of interest, and the other is noise from confused or uncooperative respondents. Additionally, the signal component may be a mixture; for example, women and men could show different patterns of response. Similarly, the noise component is often a mixture; data quality problems can entail several different patterns of error (cf. Karr et al. 2006), and these may be characterized through a mixture model.

More generally, mixture models arise whenever observations are drawn from multiple populations or processes, but there is no observable quantity to indicate which population or process produced a specific observation. For example, in the context of cluster analysis, suppose there are \( K \) multivariate distributions \( F_1, \ldots, F_K \); let \( f_k(x | \theta_k) \) denote the density function of the \( k \)th distribution, where \( \theta_k \) is the associated parameter. An observation is obtained by selecting the \( k \)th distribution with probability \( \pi_k \), where \( \pi_1 + \cdots + \pi_K = 1 \), and then drawing a random value from it. The analyst observes \( X \), but does not know from which
distribution it was taken. In this case, the density of \( X \) is

\[
f(x) = \sum_{k=1}^{K} \pi_k f_k(x | \theta_k).
\] (1.1)

Given a sample of such observations, and depending upon what assumptions are tenable, one wants to estimate some or all of \( K, \pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K \) and the forms of the distributions \( F_1, \ldots, F_K \). Additionally, one may want to estimate the probability that a specific observation was drawn from each of the distributions (i.e. that the observation belongs to a specific cluster).

In practice, there is great difficulty in deciding upon the tenable assumptions. Both Bayesian and frequentist methods require complete specification of the model, and even minor mistakes in that step can derail the analysis. Often one would prefer a procedure that allows exploratory analysis, with sequential extraction of the simplest and/or largest and/or most interpretable components, followed by re-examination of the data.

To that end this paper describes a ‘cherry-picking’ algorithm. The procedure is computer-intensive. It identifies trial components based on a few observations, and then selects (cherry-picks) other observations that best fit that trial component. If the resulting set of all selected observations is relatively large and has relatively good internal fit (in a context-specific sense that is discussed later), then the set is removed from the dataset for separate analysis.

This cherry-picking heuristic is not entirely novel. It was implicit in the elemental sets method proposed by Theil (1950), and further developed by Rousseeuw (1984) and Hawkins et al. (1984) to handle outliers in regression. Somewhat similar work has recently been done in the context of computer vision by Chen & Lerman (2009). A related idea, in the context of contingency tables, was proposed by Li (2002). In terms of robustness, cherry-picking is also related to S-estimation, in which one selects the minimum volume region that contains a specified fraction of the data, and then fits the model to those data (Rousseeuw & Yohai 1984; Hawkins 1993b). Our contribution is to extend the scope beyond outlier detection, using it as a general tool for handling mixture sparsity.

Section 2 introduces the algorithm and describes its application in the context of regression. The method is illustrated for low and moderate dimensions using a combination of simulation and data on properties of Wikipedia articles. Section 3 applies the algorithm to cluster analysis, using keystroke data collected to study typing rhythms as a method for user identification. Section 4 considers multidimensional scaling, and shows the robustness that is possible in the presence of artificially corrupted geodesic data. Section 5 draws brief conclusions.

2. Regression

Mixture models arise in multiple regression when an observation \( Y_i \) with covariates \( x_i \in \mathbb{R}^p \) may follow one of \( K \) possible linear relations. Under the classic assumptions (independent Gaussian errors, and the covariates \( \{x_i\} \) are measured without error), the model in equation (1.1) extends as

\[
f(y) = \sum_{k=1}^{K} \pi_k \phi(x' \beta_k, \sigma^2_k),
\] (2.1)

Phil. Trans. R. Soc. A (2009)
where $\phi(a, b)$ is the density of the normal distribution with mean $a$ and variance $b$. The analyst would like to estimate $K$, the regression coefficients $\beta_k$ and the regression variances $\sigma_k^2$.

This section describes the Bayesian mixture model analysis and the cherry-picking algorithm in the context of simple linear regression, and then extends the comparison with higher dimensions and the analysis of two datasets.

(a) Simple linear regression

Suppose one has 50 points that are generated as a mixture of three components.

(i) With probability 0.4, an observation comes from the linear regression $Y = x + \epsilon$ where $\epsilon \sim N(0, 0.04)$.

(ii) With probability 0.3, an observation comes from the linear regression $Y = 1 - x + \epsilon$, where $\epsilon \sim N(0, 0.09)$.

(iii) With probability 0.3, an observation is uniformly generated in $[0,1]$.

All $x$ values are generated uniformly on $[0,1]$. We analyse such data using the cherry-picking algorithm and compare it with the results of a Bayesian mixture model.

The heuristic behind the cherry-picking algorithm is that, if one is able to choose just three points that come from one of the two lines, then one can probably collect nearly all of the other points generated from that line by adding points that lie near the line fitted to the initial three observations. Obviously, this will sweep in some extraneous points, either from the uniform component or from the region near the intersection of the two lines. Additionally, instead of just relying upon the line estimated from the first three points, it is better to adaptively refit the line as one accumulates more data.

However, if one does not choose three points from the same line, then, with high probability, one of two things could happen, both of which are diagnosable as a failure in the initial selection. First, it could be that the line fitted to the three chosen points has large residual variance; in that case, nearly all of the other points would be swept up, no matter which component generated them, because their residuals are similar to those from the initial choice. However, this situation is flagged by both the large residual variance and the inclusion of nearly all points in the final set. Second, it could be that the line fitted to the three chosen points has, by chance, small residual variance, even though it includes data from more than one component. In this case, the problem is detected because the number of points that can be swept up will be relatively small.

More specifically, the implementation of the cherry-picking algorithm for a mixture of simple linear regressions is as follows.

Step 1. Select a set of three observations at random. Fit a line and calculate $R^2$, the conventional measure of variability in the response explained by the regression.

Step 2. Extend the initial set of three observations by adding all data points that do not decrease the current value of $R^2$. 
Step 3. Fit the line to this extended set, and add all observations whose residuals lie within an 80 per cent confidence band on the line. Refit the line again, and add all observations that lie within an 80 per cent confidence band. Repeat until no new observations are added.

Step 4. If the number of included observations exceeds 20 per cent of the data, declare the fitted line as an estimate of a component. Remove those data.

Step 5. Return to Step 1 and repeat until no new component of reasonable size is found.

This is an exploratory procedure, and if one has information about the kinds of structures in the data (e.g. the regression variance may be available if the precision of the measurement instrument is known), one could use that in tuning the analysis.

One could accelerate this algorithm by generating, say, 1000 initial triples of data, and inspecting the distribution of the $R^2$ values to determine cutpoints that would enable early termination of the subsequent search. This would forestall search when the initial triple had an $R^2$ that was too small. Less obviously, it also gains speed by terminating triples that have a very large $R^2$; here the points happen, by chance, to fall almost perfectly upon a line, and the cherry-picking algorithm will require many iterations to sweep in suitable points in Steps 2 and 3. For the examples in this paper such acceleration is not needed, but the strategy is helpful in other settings.

Step 2 selects all of the data that cluster tightly around the line defined by the initial triple. If the triple comes from a single component, this should constitute a substantial fraction of the data; otherwise, the fraction will generally be small. Step 3 iteratively refines the model estimated from the selected data, adding in additional data that seem likely to belong to the component being fitted. The choice of an 80 per cent confidence band as the inclusion criterion is a reasonable guess, but it could be tuned as needed. Its main impact is a slight downward bias in the estimate of the regression variance.

Step 4 identifies and removes the data corresponding to a fitted component, and Step 5 restarts this sequential structure extraction process with the remaining data. To avoid the rare mistake of removing a spurious component, one might repeat Steps 1–4 several times, and then identify and remove only the component accounting for the largest fraction of the data among those repetitions.

We compare this cherry-picking algorithm with a Bayesian mixture model analysis. One needs priors on the number of components, the component probabilities, the regression coefficients, and the regression variances; the description below refers to the notation in equation (2.1). For this example, we simplify (and assist) the analysis by assuming it is known that there are $K = 3$ components. The prior on the component probabilities is Dirichlet with parameter $(1,1,1)$. The priors on the three regression vectors $\beta_k$, $k = 1, 2, 3$, are independent of one another and each is $N(0, I)$. The priors on the three regression variances $\sigma_k^2$ are also independent of one another and each has a gamma distribution with parameters $\alpha = 2.0244$ and $\beta = 0.5122$ (so that the mean and variance of the prior are 0.5 and 10, respectively). Collectively, these priors are relatively diffuse, but reasonably centred upon the true values. One uses Gibbs sampling to find posterior distributions.

 Phil. Trans. R. Soc. A (2009)
Figure 1. The figure shows nine simulated datasets composed of a mixture of three components. Two components are linear \( (Y = X \text{ with } N(0,0.04) \text{ noise and } Y = 1 - X, \text{ with } N(0,0.09) \text{ noise}) \) and the third is uniformly generated in \([0,1]\). The solid lines indicate the two true linear components \( Y = x + \epsilon \text{ and } Y = 1 - x + \epsilon \). The dashed lines indicate the fitted cherry-picking regressions, and the dotted lines are the Bayesian mixture model regressions. Points plotted as triangles are generated from the first component, points plotted as circles are generated from the second component, and points plotted as plus signs are generated from the uniform component. Visually, this indicates that the cherry-picking algorithm is competitive with Bayesian mixture modelling.

Note that the uniform component is not explicitly modelled as such. In principle, one expects it to be approximated by a line with zero slope and variance of about \( \frac{1}{12} \) (matching to the uniform distribution) but the normal model for error will fit it poorly. This reflects the practical difficulty in modelling all of the components correctly. It also puts the comparison on a fair footing, since the cherry-picking approach does not model the uniform component either.

Figure 1 gives a visual comparison of the cherry-picking algorithm with Bayesian mixture modelling for nine datasets generated under our scheme. In general, cherry picking does a better job of finding the mixture components—the Bayesian mixture model appears to have difficulty in estimating the regression variance, leading to markedly incorrect estimates.
Table 1. Results from 1000 simulations that compare Bayesian mixture modelling (BMM) with the cherry-picking algorithm on 50 observations generated from a three-component mixture. The true values of regression parameters for the first and second components are shown; the third component is a uniform distribution.

<table>
<thead>
<tr>
<th>parameter</th>
<th>truth</th>
<th>BMM</th>
<th>cherry-picking</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_1 )</td>
<td>0.4</td>
<td>0.34</td>
<td>0.20</td>
</tr>
<tr>
<td>( \beta_0^{(1)} )</td>
<td>0</td>
<td>0.27</td>
<td>-0.47</td>
</tr>
<tr>
<td>( \beta_1^{(1)} )</td>
<td>1</td>
<td>0.28</td>
<td>1.89</td>
</tr>
<tr>
<td>( \sigma_1^2 )</td>
<td>0.04</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>0.3</td>
<td>0.34</td>
<td>0.37</td>
</tr>
<tr>
<td>( \beta_0^{(2)} )</td>
<td>1</td>
<td>0.53</td>
<td>1.00</td>
</tr>
<tr>
<td>( \beta_1^{(2)} )</td>
<td>-1</td>
<td>-0.26</td>
<td>-1.10</td>
</tr>
<tr>
<td>( \sigma_2^2 )</td>
<td>0.09</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>( \pi_3 )</td>
<td>0.3</td>
<td>0.32</td>
<td>0.52</td>
</tr>
<tr>
<td>( \beta_0^{(3)} )</td>
<td>0.39</td>
<td>0.009651</td>
<td>0.43</td>
</tr>
<tr>
<td>( \beta_1^{(3)} )</td>
<td>0.01</td>
<td>0.018248</td>
<td>0.19</td>
</tr>
<tr>
<td>( \sigma_3^2 )</td>
<td>0.02</td>
<td>0.000006</td>
<td>0.08</td>
</tr>
</tbody>
</table>

These analyses were repeated 1000 times. Table 1 shows the mean and standard deviation for the parameters in the three components (to avoid label-switching problems, estimated components were assigned to the component in the true mixture model that provided the best match). The cherry-picking procedure performs better than Bayesian mixture modelling in terms of estimating the regression parameters, but not as well in terms of estimating the component weights \( \pi_i \) (which is not a surprise—the first component to be extracted pulls in points near the intersection).

The analysis summarized in table 1 attempted to extract the uniform component by fitting a line. This implementation reflects the reality that, in practice, one does not know the number or form of the components. Extensive preliminary experience shows that, when trying to extract linear regression signal from uniform noise, one tends to find one large component with intercept near \( \bar{Y} \), slope near 0 and large variance.

As an example of the cherry-picking algorithm in simple regression, we consider an application to Wikipedia data. Wikipedia is an on-line encyclopædia with content that has been developed through open collaboration. Anyone may contribute, which raises concerns about accuracy. Therefore Wikipedia maintains strict version control, with a record of all changes that have been made. Researchers use those data to study models of change in a complex network; our data come from http://download.wikimedia.org/, and were archived on 1 January 2007.
Figure 2. A plot of the number of revisions of a Wikipedia entry against the size of the entry in bytes. Superposed on the data plot are five lines identified by the cherry-picking algorithm. Different symbols distinguish the different lines; the triangles that point upwards denote points that were not assigned to any of the five lines.

In particular, there is interest in how the number of edits that a Wikipedia page receives is related to the length of the page (in bytes). Figure 2 plots this, based upon a ‘thinned’ sample of 4710 entries from the full Wikipedia version control data (we took every 1300th entry). As shown, the data appear to represent a mixture of several different linear relationships. The five lines found by the cherry-picking algorithm are also shown, and these lines are visually satisfactory in describing the plot.

We believe that these lines identify different kinds of Wikipedia pages. Several kinds of pages are used for distinct content management purposes; one set catalogues new entries that are added under a Wikipedia topic. For these pages, content changes rapidly, and each change increments the page size with a very short pointer. Other pages are content pages, with a relatively low rate of revision, or annotated lists, with an even lower rate of revision. Further work is required to confirm this, but the interpretation accords with related work by Warren et al. (2008). Some people have proposed analysing these data on the log scale, but that would be inappropriate if, in fact, the raw data are a mixture linear models, as domain knowledge suggests.

Table 2 shows the estimated values of the five regression lines. As a matter of applied statistics, we note that it would probably be better to use a robust regression, rather than ordinary least squares, since the plot shows that the data contain some high-leverage points. Use of robust regression would be computationally difficult in mixture modelling, but it is quite simple with a cherry-picking algorithm.
Table 2. The regression parameters found by the cherry-picking algorithm applied to Wikipedia data.

<table>
<thead>
<tr>
<th>parameter</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>0.42302</td>
<td>0.55150</td>
<td>0.00234</td>
<td>0.01932</td>
<td>0.00191</td>
</tr>
<tr>
<td>$\beta_0$</td>
<td>1.00229</td>
<td>4.24518</td>
<td>-29.19885</td>
<td>76.52860</td>
<td>198.40380</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>0.00003</td>
<td>0.00351</td>
<td>0.00124</td>
<td>0.00765</td>
<td>0.02175</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.00955</td>
<td>287.42573</td>
<td>442.95502</td>
<td>2807.07285</td>
<td>4560.11609</td>
</tr>
</tbody>
</table>

(b) *Multiple linear regression*

The cherry-picking algorithm applies in higher dimensions. The main difficulty in extending it to multiple regression is that, as the dimension increases, the size of the starting subsample must also increase, since $p + 2$ points are needed to calculate a measure of goodness-of-fit. This means that the chance of selecting a starting subsample that is drawn entirely from one of the components gets small (approximately $\pi^{p+2}$, where $\pi$ is the probability of the most likely component). Thus more iterations are needed before a sensible component is discovered, and, at a certain point, the method becomes infeasible unless one has partial information on component membership (as happens in semisupervised learning; cf. Blum & Mitchell 1998).

We consider a simulation to study the performance of the cherry-picking algorithm for data in $\mathbb{R}^5$. This is not very high-dimensional, but it is sufficiently large to have values in practice. We suppose one has 100 points that are generated as a mixture of three components.

(i) With probability 0.4, an observation comes from the regression equation

$$Y = x_1 + 2x_2 + 4x_3 + \epsilon$$

where $\epsilon \sim N(0, 0.01)$.

(ii) With probability 0.4, an observation comes from the regression equation

$$Y = -1 - x_3 - 2x_4 - 4x_5 + \epsilon,$$

where $\epsilon \sim N(0, 0.009)$.

(iii) With probability 0.2, an observation is uniformly generated in [0, 1].

As before, all $x$ values are generated uniformly on [0, 1].

The cherry-picking algorithm is essentially the same as for simple linear regression, but with a few modifications. First, the initial sample size is now 7 rather than 3. Second, we require that the range in the explanatory variables be at least 0.35 in one or more of the five dimensions; this ensures that the starting subsample is dispersed, and avoids instabilities that can arise when the initial subsample is ‘clumped’. Third, if an initial subsample has root mean squared error greater than 1, then it is immediately discarded (this is done to accelerate the computation; it is not essential, but such methods become increasingly valuable as the dimension increases).

Table 3 shows the results of the simulation. The estimates are extremely accurate, especially when compared with the results from simple linear regression. Presumably this is explained by the larger sample size and the smaller regression variances.
Table 3. Results from an analysis of 100 observations from a three-component mixture model for multiple regression.

<table>
<thead>
<tr>
<th>component parameter</th>
<th>truth</th>
<th>estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_1 )</td>
<td>0.40</td>
<td>0.44</td>
</tr>
<tr>
<td>( \beta_1' )</td>
<td>(0, 1, 2, 4, 0, 0)</td>
<td>(−0.17, 1.06, 2.05, 4.12, −0.06, 0.07)</td>
</tr>
<tr>
<td>( \sigma_1^2 )</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>0.4</td>
<td>0.39</td>
</tr>
<tr>
<td>( \beta_2' )</td>
<td>(−1, 0, 0, −1, −2, −4)</td>
<td>(−1.02, 0.03, −0.03, −1.06, −2.02, −3.98)</td>
</tr>
<tr>
<td>( \sigma_2^2 )</td>
<td>0.009</td>
<td>0.008</td>
</tr>
<tr>
<td>( \pi_3 )</td>
<td>0.2</td>
<td>0.17</td>
</tr>
<tr>
<td>( \hat{\beta} )</td>
<td>uniform on (0, 1)</td>
<td>(0.85, −0.06, −0.36, 0.07, −0.37, −0.04)</td>
</tr>
<tr>
<td>( \sigma^2 )</td>
<td></td>
<td>0.05</td>
</tr>
</tbody>
</table>

This simulation used multiple regression because it is a natural benchmark for performance. The cherry-picking algorithm is a plug-in procedure, so one could substitute a non-parametric regression or a variable selection technique, such as the LASSO.

The analyses in this section used \( R^2 \) as a measure of fit. This measure could be replaced by other possibilities; e.g. minimizing the sum of the absolute deviations, or incorporating a complexity penalty on the fit such as Mallow’s (1973) \( C_p \) statistic or Akaike’s (1973) information criterion. The cherry-picking algorithm extends immediately to such alternatives. The only caveat is that the measure of lack-of-fit should not depend strongly upon the sample size. If the starting-point subsample must be large, then the chance of finding a subsample drawn from a single component of the mixture quickly becomes small.

### 3. Cluster analysis

Traditional cluster analysis tends to perform poorly in high dimensions; a few outliers can distort the analysis (Kaufman & Rousseeuw 1990), and the Curse of Dimensionality makes it difficult to discover the hidden structure (Hall et al. 2006; Bickel & Levina 2008). For these reasons, we apply the cherry-picking heuristic to the problem of multivariate cluster analysis.

In multivariate cluster analysis, it often happens that a cluster is well defined by a relative handful of the variables, but that different clusters are distinguished by different sets of variables. Thus most or all of the variables are needed to specify all of the clusters, but, to distinguish a particular cluster, the problem is locally low-dimensional. This is distinct from the case treated by Kim et al. (2006), who used a Dirichlet process mixture model to select, from among a large number of variables, the unknown small subset that was sufficient to identify all of the clusters.

An example of locally low-dimensional clustering arises in biometric identification of typists from patterns in their keystroke hold times and inter-key latencies. This application is important in computer security, where one hopes...
Cherry-picking for complex data

One study of keystroke patterns (Killourhy & Maxion 2008) had 51 typists type the same ten letter password (‘.tie5Roanl’, chosen because it contains both awkward and common combinations on the qwerty keyboard, and a shift, exercising a range of typing dynamics). The participants typed the ‘strong’ password 400 times, in eight sets of 50. Each typing of the password produced a vector in $\mathbb{R}^{21}$, consisting of eleven hold times (one for each of the ten letters and one for the return key typed at the end of the password) and ten keydown–keydown latencies (one for each pair of consecutively typed keys).

Good biometric identification is achieved if the different typists show strong clustering. However, it was suspected that the characteristic patterns of a typist might show up clearly in only some of the components of the vector, and that those components would differ according to the typist. For example, a fast hold time on the period key might be characteristic of one typist, while a slow keydown–keydown latency between the L and return keys might be characteristic of another.

For clarity of description, we focus upon clustering the first five typists in the database, rather than all 51. This focus also allows us to separate our exploration of multidimensional clustering from the problem of clustering many (51) typists at once. Additionally, to minimize learning effects, we use only the data from the last of the eight sets of repetitions. Thus we have 50 observations in $\mathbb{R}^{21}$ on each of five typists. Figure 3 compares the parallel-coordinate plots (cf. Wegman 1990) for each of the five typists.

(a) Algorithm for cherry-picking multivariate clusters

Our analysis is an example of unsupervised learning, in that we do not use information on the identities of the typists. Instead, we do cluster analysis, and assess its performance according to whether it produces five well-defined clusters that contain nearly all of the data, and such that each cluster consists almost entirely of vectors from exactly one of the typists.

To use the cherry-picking approach in cluster analysis, one must draw an initial subsample that (by chance) consists entirely of members of a single cluster. From that initial subsample, one generates a rule that determines which other observations are sufficiently nearby that they should be added to that cluster—typically, this will entail estimating a covariance matrix that defines a Mahalanobis distance, but in high dimensions this must be combined with variable selection. Then one sweeps in all data that satisfy the rule, adaptively updates the rule with the new data and continues. If the initial sample includes points from multiple clusters, then the rule that is generated will be too inclusive, and the typical result is that superclusters, containing multiple clusters, will be formed. This is diagnosable by repeating the procedure on the supercluster, or by noticing that the cluster that is formed is unreasonably large (the approach used herein).

The key challenge in implementing this approach lies in the handling of high-dimensional data. If one estimates a full covariance matrix for $p$-dimensional data, one requires $\binom{p}{2}$ observations. This implies that the initial sample has to be
Figure 3. The parallel-coordinate plot depicts the typing rhythms of each of the five typists. Each typist’s timing components are shown in a separate panel so that their styles can be compared and contrasted.

large, and thus the probability that it contains only observations from the same cluster will be small. Furthermore, in our example, $p = 21$ so one would need 210 observations to estimate the full covariance matrix; but there are only 50 observations per typist.

To address this problem we cherry-pick two variables from among the 21, based on their univariate variances in the initial random subsample, from which we estimate a Mahalanobis metric that is used to sweep in other nearby observations. Specifically, we choose an initial sample of size 3 at random (which has probability $49 \times 48/(249 \times 248) = 0.038$ of coming from the same typist). From that sample, we could select the two variables that have the smallest variance among the three observations, but for this application we actually pick the two that have the smallest ratio of the sample standard deviation to the sample mean. This is done because the hold times and latencies for keystroke data tend to be right-skewed, reflecting the fact that there is a physical lower limit to typing speed, but no such limit on delays. Using the two selected variables and their covariance, we have a crude estimate of Mahalanobis distance that is improved as more data are swept in and the bivariate distance matrix is updated.

Specifically, the steps in the algorithm are as follows.

**Step 1. Seeding:** Three observations are randomly chosen as a cluster seed. For this seed sample, a measure of dispersion is calculated for each of the 21 variables (i.e. the ratio of the sample standard deviation to its mean). The two variables with the smallest dispersions according to this measure are identified.
Table 4. A summary of the clustering of the keystroke data. The two variable columns list which variables the cluster selected. The final five columns break down each cluster into which typists’ observations were included in the cluster. In each case, the cluster roughly corresponds to data from a single typist.

<table>
<thead>
<tr>
<th>cluster</th>
<th>variable 1</th>
<th>variable 2</th>
<th>breakout by typist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hold (shift r)</td>
<td>hold (l)</td>
<td>1 48 0 1 0</td>
</tr>
<tr>
<td>2</td>
<td>hold (period)</td>
<td>latency (1 − return)</td>
<td>2 1 1 46 0</td>
</tr>
<tr>
<td>3</td>
<td>latency (n − 1)</td>
<td>hold (l)</td>
<td>0 0 44 0 6</td>
</tr>
<tr>
<td>4</td>
<td>latency (five − shift r)</td>
<td>hold (shift r)</td>
<td>2 0 4 3 41</td>
</tr>
<tr>
<td>5</td>
<td>hold (five)</td>
<td>hold (l)</td>
<td>45 1 1 0 3</td>
</tr>
</tbody>
</table>

Step 2. Selection: Using only these two low-dispersion variables, the bivariate covariance matrix for the clustered observations is calculated. For all the observations not yet in the cluster, the squared Mahalanobis distance to the mean of the cluster is calculated. Each distance is compared with the 99th percentile of the $\chi^2$ distribution (with two degrees of freedom), and those observations with distances below this threshold are added to the cluster. For intuition, the threshold is chosen so as to include nearly all (99%) of the observations that belong to one typist but to exclude outliers and most observations from other typists.

Step 3. Termination: The iteration procedure in Step 2 is repeated until the cluster converges (i.e. no new elements are added to the cluster). The size of the resulting cluster is compared with the expected size of a cluster (e.g. 50±5 elements). If the sizes do not match, the cluster is discarded, and we return to Step 1 to find a better cluster. If the sizes do match, the cluster is saved and the clustered observations are removed from the dataset. We return to Step 1 to find the next cluster. The algorithm terminates when the dataset contains too few observations to form a new cluster.

This cherry-picking clustering algorithm was run on the 250-observation keystroke data. The algorithm saved five clusters, each of which happened to contain 50 observations (but this was chance, not manipulation; other runs produced clusters that contained between 45 and 55 cases). Table 4 breaks down the composition of each cluster. The clusters each correspond strongly to one of the five typists, with that (majority) typist’s observations comprising between 41 and 48 of the 50 elements in the cluster. The variables used to distinguish the clusters from the other data show that a range of variables were used; some variables were important for several clusters (e.g. the hold time for the L key) while others appear unique to a particular cluster (e.g. the hold time for the period key).

To help visualize the progress of the algorithm, figure 4 contains six panels. The first (figure 4a) depicts the keystroke data according to its principal components. The remainder show the progress of the clustering algorithm. Note that the ellipses identifying cases for inclusion are the ellipses formed at the last stage
Figure 4. The panels depict the progression of the clustering analysis. (a) The two principal components of the data. Each typist’s observations are represented by a different symbol. The panels (b–f) depict the five clusters. (b) The ellipsoid delimiting the first cluster from the rest of the data; (c) the ellipsoid for the second cluster on the remaining data, etc.
of the loop in Step 2; some cases formed in earlier stages are no longer contained within the final ellipse formed by the fully updated covariance matrix, but those cases are still included as members of the cluster. This is a tuning issue that might be further evaluated. Also, note that, because of the distance calculation in the selection step, cherry-picking is not equivariant to transformations that do not preserve Euclidean distance (e.g. the log transform).

Figure 4 shows that the first cluster (figure 4b) found 50 observations with fast hold times on the L and R keys (the latter denoted by shift r in the table and figure because it was capitalized). Nearly all these observations belonged to the second typist. The second cluster (figure 4c) found 50 observations with fast hold times on the period key and slower keydown–keydown latencies in moving from the L key to the return key. Nearly all of these data belonged to the fourth typist.

In each case, when a cluster of observations was cherry-picked by the algorithm, those elements corresponded closely with one of the five typists, and the variables relevant to the cluster reveal what is characteristic of the typist’s unique keystroking style. From the standpoint of computer security, this is a promising result; one can accurately identify typists, and the occasional mistake imposes only the cost of having the typist re-enter the password before access is granted.

(b) Results from simulated data

To explore the performance of the algorithm under controlled conditions, a simulated dataset was generated. The data consist of 50 observations in \( \mathbb{R}^{21} \) from each of five classes. Each class has two ‘signal’ variables and 19 ‘noise’ variables. The signal variables are unique to the class (i.e. the signal variables for class 1 are 1 and 11, the variables for class 2 are 2 and 12, and so on). The signal-variable data were drawn from a normal distribution with mean 1 and standard deviation 0.1, and the noise-variable data were drawn from a standard normal distribution (mean 0 and standard deviation 1).

The cherry-picking algorithm was run 1000 times on this dataset. The algorithm was configured to save a cluster when its size was within five elements of the expected cluster size (i.e. 50 ± 5 cases). Table 5 summarizes the results. Each run resulted in five clusters, and each cluster was composed of a majority of cases from one class, and a few cases from the other classes.

4. Multidimensional scaling

Robustness is a major concern when using multidimensional scaling (cf. Spence & Lewandowsky 1989). It is easy for a handful of outliers to distort the result so significantly that no useful structure is discovered. Additionally, for applications in which the proximity matrix represents subjective assessments of distances (as often arises in psychometric and marketing experiments) it is reasonable to expect that the respondent is mixing several different senses of distance. This creates exactly the sort of situation in which the analyst needs to identify large subsets of the data that are well explained by a simple model.

For example, we apply multidimensional scaling to find a low-dimensional representation of the locations of 100 major cities in the eastern United States. These points lie on the surface of a globe, and cannot embed perfectly in
Table 5. The results of running the cherry-picking algorithm 1000 times on a synthetic dataset show consistent performance. The numbers in each column indicate how many elements from each class are included in the cluster. The first nine rows show the results of individual runs. In each, five clusters were generated, each one roughly corresponding to the data points of one class. The final row shows the average results over all 1000 runs, rounded to the nearest integer.

<table>
<thead>
<tr>
<th></th>
<th>cluster 1</th>
<th>cluster 2</th>
<th>cluster 3</th>
<th>cluster 4</th>
<th>cluster 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>47</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>49</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>45</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>avg</td>
<td>46</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

two-dimensional Euclidean space. We shall distort those data by inserting outliers, in order to show how standard multidimensional scaling fails, but a cherry-picking version succeeds.

In our application, we use Kruskal–Shephard non-metric scaling (cf. Borg & Groenen 2005). This seeks a mapping of the data that minimizes the stress function,

\[
\text{stress}(x_1, \ldots, x_n) = \frac{1}{\sum_{j \neq j'} d_{jj'}^2} \sum_{i \neq i'} \left[ g(||x_i - x_{i'}||) - d_{ii'} \right]^2,
\]  

(4.1)

where the \(d_{ii'}\) are the distances between the two-dimensional embeddings of the points \(x_i\) and \(x_{i'}\) and the \(g(\cdot)\) is an arbitrary monotonically increasing function (this implies that the fit depends only upon the ranks of the interpoint distances). The fitting is done by alternating isotonic regression to find an estimate of \(g\) with gradient descent to find an estimate of the \(d_{ii'}\); our implementation used the procedure in the R software package. We use the Kruskal–Shephard algorithm, rather than simpler parametric methods, in order to emphasize how easily the cherry-picking method can be implemented when using complex modelling, and to underscore the ease with which computationally intensive procedures can be plugged into the cherry-picking algorithm.

In multidimensional scaling, a small proportion of outliers or similar data quality problems can distort the fit into uninterpretability. The minimization of stress is then driven by the outliers rather than the accurate data, leading to maps that are highly distorted representations of the true geography. But cherry-picking allows the fitting procedure to robustly ignore points that cause large increases in lack-of-fit.

The cherry-picking algorithm described previously is implemented as follows. First, one selects a random starting-point subsample. In this case, that subsample contains four points, in order to ensure that the embedding from \(\mathbb{R}^3\) to \(\mathbb{R}^2\).
generates a measure of lack-of-fit, or stress. Then we sweep up all the points whose inclusion does not increase the overall stress per observation, recalculate the embedding, and repeat until no new points may be added. If this final set of points constitutes a scientifically meaningful fraction of the data, these points are removed. If not, then a new starting-point subsample is drawn at random and the process is repeated.

There are various ways to speed up the calculation. For example, in this application to inter-city distances, we have a sense of how much distortion the curvature of the Earth should cause, and could exploit this by immediately discarding any initial subsample whose stress was too large.

To study the performance of the cherry-picking algorithm on these data, we simulated three datasets from the inter-city distances by contaminating the data with varying amounts of error. Specifically, we chose \( 100(1 - q) \) pairs of data points at random and added 50 to the corresponding distance measures, for \( q = 0.9, 0.8 \) and 0.7. The total stress measures for these contaminated datasets are 9.73, 58.22 and 41.25, respectively, whereas the stress for the original, clean dataset equals \( 8.42 \times 10^{-12} \). Figure 5 shows multidimensional scaling representations of the correct data and its three, increasingly corrupted, versions.

Table 6 summarizes the results of the cherry-picking algorithm applied to these four datasets. It shows that, as the data become more corrupted, fewer cities are swept into the multidimensional scaling analysis, but the fitted stress of those that are included is in conformity with the magnitude of the stress expected from uncorrupted inter-city distances.

Figure 6 shows how the stress increases for the situation in which 30 per cent of the inter-city distances are corrupted. After the first 76 of the 100 cities are swept in by the algorithm, there is a dramatic increase in the average stress. This kind of plot is a useful tool in deciding when to terminate the sweeping-in step.

The inter-city distance dataset emphasizes two features of the cherry-picking algorithm. First, it has significant potential for problems in which data quality is an issue. Although one could try to describe this problem as a mixture model, in which an unknown fraction of the data shows a low-stress solution and a different fraction does not, it is difficult to imagine that it would be simple to fit such a model. In contrast, the cherry-picking algorithm is quite transparent. A second point, that interacts with the first, is that, although the Kruskal–Shephard non-metric scaling measure of stress in equation (4.1) is relatively more complex and computationally intensive than alternative definitions of stress, it was entirely straightforward to plug this measure into the cherry-picking heuristic. The ease resulted from the fact that the cherry-picking allows the user to employ almost any multidimensional scaling method that is wanted; the complexity of the method may slightly slow the cherry-picking algorithm, but it does not affect the underlying logic.

5. Discussion

Modern computer-intensive statistics fits complex models, but complex models are so flexible that they can be strongly influenced by outliers. The problem is exacerbated in multivariate settings, and quickly grows worse as the dimension

Phil. Trans. R. Soc. A (2009)
increases. This paper examines cherry-picking as a way to exclude outliers when fitting models; it applies when the data exhibit mixture sparsity. Such situations arise when the observed data contain many different kinds of structure (e.g. multiple regression lines, or clusters of different shapes and locations).

To address such problems, we describe a procedure that searches among the sample to find a large subset for which a statistical model can be fitted well. Thus we 'cherry pick' the best data for the model of interest, minimizing the chance that bad data will distort the inference. This procedure has surprising generality, and applies to many different kinds of inference, including regression, cluster analysis and multidimensional scaling. One can apply the method iteratively, so that the first iteration fits a model to the largest portion of the data that is consistent with the model, then one removes that fitted data and uses the technique again to fit a new model to the second largest portion of the data. Proceeding in this way, one
Cherry-picking for complex data

Figure 6. A plot of the stress, as measured by equation (4.1), that arises from serial inclusion of cities into the analysis. In this example 30 per cent of the inter-city distances are distorted, and cities are added in order of increasing stress. Note that 76 cities are added without significantly increasing the stress, but after that point the effect of distorted data is obvious. This kind of plot would help researchers in deciding how many observations to include in a cherry-picking analysis.

Table 6. Comparison of results from three data quality scenarios. As a larger fraction of the data are corrupted with large errors, fewer cities are included in the embedding but those that are included show stress levels consistent with the uncorrupted data.

<table>
<thead>
<tr>
<th>fraction corrupted</th>
<th>number of cities swept in</th>
<th>final stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>80</td>
<td>$5.03 \times 10^{-12}$</td>
</tr>
<tr>
<td>0.2</td>
<td>83</td>
<td>$5.71 \times 10^{-12}$</td>
</tr>
<tr>
<td>0.3</td>
<td>76</td>
<td>$4.91 \times 10^{-12}$</td>
</tr>
</tbody>
</table>

can extract different kinds of structure in an automatic way. In particular, this means that one does not need to ‘seed’ the fit with extensive model specification, as is required for traditional mixture modelling. Also, one can easily explore alternative kinds of parametric substructures (linear, nonlinear and so forth).

In some problems, the algorithm does not extend well to large dimensions. In particular, for regression analysis or cluster analysis with clusters determined by many covariates, the starting-point subsamples may need to be so large that the chance of finding a subsample that is drawn entirely from one component of the mixture becomes very small. On the other hand, in applications such as multidimensional scaling, the relevant dimension is that of the space in which the observations are to be embedded, and then the dimension of the original data is not problematic.

One of the attractive features of the cherry-picking procedure is its plug-in capability. With very little work, one can replace ordinary least squares regression by a robust or non-parametric regression, or a regression method that is designed for variable selection (which can partially address the difficulty the cherry-picking
algorithm encounters with high-dimensional data sets). Similarly, Mahalanobis distance cluster analysis can be replaced by any other standard clustering rule, and in multidimensional scaling one can use simpler and more traditional stress measures than the non-metric scaling employed in this paper.

The procedure we describe extends easily to many statistical applications, requiring only some measure of fit. There is a natural graphical tool for determining when one has found essentially all of the good data, and that tool performed well in the studies reported here.

This research was supported in part by the Isaac Newton Institute for Mathematical Sciences, Cambridge, UK, by US National Science Foundation grants CNS-0430474 and CNS-0716677, and by the US Army Research Office grant DAAD19-02-1-0389.

References