Finite discrete Markov process clustering

Greg Ridgeway
t-gregr@microsoft.com
greg@stat.washington.edu

September 4, 1997

Technical Report
MSR-TR-97-24

Microsoft Research
Advanced Technology Division
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract

In this paper I describe a probabilistic method for clustering discrete Markov processes with a pre-specified number of clusters. I derive two algorithms for estimating the model parameter; one based on a Gibbs sampler and the other based on an EM algorithm. The Gibbs sampling algorithm is accurate at the expense of speed and a constrained EM algorithm is optimized for speed at the expense of accuracy. A hybrid algorithm is also formulated.

1 Problem description

Consider an s-state discrete Markov process where the transition matrix for the process, $\underline{T}$, is unknown. Now assume that we believe that the process came from one of m distinct transition matrices so that,

$$\underline{T} = \sum_{l=1}^{m} \delta_l P_l \qquad \textbf{(1)}$$

where $\delta_\ell=1$ if the process came $\underline{P}_\ell$ and otherwise $\delta_\ell=0$. For example we may observe a set of N processes with states enumerated by 1, ... , s.

Process 1: 4 7 3 4 2 5 4 3 2 1 5
Process 2: 5 3 7 9 1 2
Process 3: 3 4 1 3 2 1 2 3
:
Process N: 4 1 8 7 4 3 2 1

From the start we do not know which of the m transition matrices generated each process, nor do we know the initial state distribution, nor the transition probabilities in the transition matrices. Despite our lack of knowledge, estimation is reasonably tractable. In the Bayesian clustering framework the vector $\underline{\delta}$ is the unobserved class variable. If we have a collection of processes generated from $\underline{T}$ then we can compute joint posterior densities for the parameters in this model. Of particular interest are the elements of the $\underline{P}_\ell$ and the class variable.

This discrete Markov process clustering methodology is a potentially useful method for clustering graph traversals, discovering task-related sequences, and next-step prediction.

1.1 Notation

The following notation will be used throughout this report.

i - index for the origin states
j - index for the destination states
k - index for the observed processes
ℓ - index for the clusters
N - the number of observed processes
i_0 - the initial state of the process
n_{ij} - the number of times a process transitioned from state i to state j
m - the number of clusters
s - the number of states
p_ℓ - a probability vector of length s specifying cluster ℓ's initial state distribution
P_ℓ - an $s \times s$ matrix of transition probabilities for cluster ℓ.
$\ell \to k$ - indicates the event that cluster ℓ generated process k. This implies that P_ℓ is the transition matrix underlying this process. $\Pr(\ell \to k | \underline{\delta}^{(k)})$ is the ℓth element of $\underline{\delta}^{(k)}$ and $\Pr(\ell \to k)=\alpha_\ell$.

$\underline{\alpha}$ - a probability vector of length m that contains the proportion of processes coming from any particular cluster.

$\underline{\delta}^{(k)}$ - the probability vector containing the probability that cluster ℓ generated process k. The Gibbs sampling algorithm (section 2.1) and the constrained EM algorithm (section 3.2) force this vector so that the ℓ^{th} element is 1 if process k was generated from P_ℓ and all other elements are 0.

2 Gibbs sampling algorithm for process clustering

Under a Bayesian formulation where a prior distribution of the parameters can be specified the likelihood density can be inverted by Bayes' Theorem in order to obtain a posterior density of the model parameters. Although the posterior distribution can be quite complex, Gibbs sampling provides a means for exploring the posterior distribution.

2.1 Likelihood function

$\underline{P}_\ell$ is a Markov transition matrix.

$$P_l = \begin{bmatrix} {}_{(l)}P_{11} & \cdots & {}_{(l)}P_{1m} \\ \vdots & \ddots & \vdots \\ {}_{(l)}P_{m1} & \cdots & {}_{(l)}P_{mm} \end{bmatrix} \qquad \textbf{(2)}$$

N is the number of observed processes and n_{ij}'s indicate the observed number of transitions that process k made from state i to state j. Therefore the likelihood is as follows

$$f(\underline{n} \mid \underline{p}, \underline{P}, \underline{\delta}) = \prod_{k=1}^{N} \prod_{l=1}^{m} \left({}_{(l)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}} \right)^{\delta_l^{(k)}} \qquad \textbf{(3)}$$

2.2 The algorithm

Specifying uninformative priors in this case is quite natural. Let $\underline{\alpha}$ be vector of length m of the mixture proportions. Then $\underline{\delta}$ will be distributed multinomial(1,$\underline{\alpha}$). A Dirichlet($\underline{1}$) hyperprior for the hyperparameter $\underline{\alpha}$ corresponds to a uniform prior over the m-dimensional simplex. Similarly, every row in every transition matrix will also have a Dirichlet prior over the s-dimensional simplex. In summary, the prior distributions are

$$\begin{aligned} &\delta^{(k)} \sim \text{Mult}(1, \underline{\alpha}) \\ &\underline{\alpha} \sim \text{Dirichlet}(\underline{1}_m) \\ &{}_{(l)}p \sim \text{Dirichlet}(\underline{1}_s) \\ &{}_{(l)}P_{i\bullet} \sim \text{Dirichlet}(\underline{1}_s),\ \text{where } {}_{(l)}P_{i\bullet} \text{ is the i}^{\text{th}} \text{ row of } {}_{(l)}\text{P}. \end{aligned} \qquad \textbf{(4)}$$

The joint distribution is proportional to the likelihood multiplied by the prior densities and easily follows.

$$f(\underline{p}, \underline{P}, \underline{\delta}, \underline{\alpha} \mid \underline{n}) \propto \prod_{k=1}^{N} \prod_{l=1}^{m} \left({}_{(l)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} P_{ij}^{n_{ij}^{(k)}} \right)^{\delta_l^{(k)}} \bullet \prod_{k=1}^{N} \prod_{l=1}^{m} \alpha_l^{\delta_l^{(k)}} \bullet 1 \bullet 1 \bullet 1 \qquad \textbf{(5)}$$

Assuming that the first order Markov assumption is correct, this distribution captures all of the information about the process clustering that is contained in the data. However, this distribution is rather complex and all of the usual distribution summary values (mean, variance, etc.) are extremely difficult to extract.

Appealing to a Markov Chain Monte Carlo approach to sample from this distribution can avoid this problem with a degree of computational cost.

Markov Chain Monte Carlo algorithms provide a method for drawing from complicated distribution functions. The form of the posterior distribution lends itself to a class of MCMC algorithms known as Gibbs samplers. Implementations of a Gibbs sampler partition the parameter space into blocks or sets of parameters where drawing from the distribution of the block given all of the other blocks is simple. Iterations of the Gibbs sampler in turn draw new values for each block of parameters from these block conditional distributions. An ergodic theorem assures us that the Gibbs sampler will produce a sample from the target distribution function.

2.3 Block conditionals

The parameter space of the process clustering problem at hand partitions naturally. The rows of every transition matrix, the vector α, and each δ will be block updated. The block conditionals are easily found from the posterior shown in (5).

$$\begin{aligned}
f\left({}_{(l)}p \middle| {}_{(l)}p^{-}, \underline{n}\right) &\propto \prod_{k=1}^{N} {}_{(l)}p_{i_0^{(k)}}^{\delta_l^{(k)}} \\
&= \prod_{k=1}^{N}\prod_{i=1}^{s} {}_{(l)}p_i^{\delta_l^{(k)}\bullet \mathrm{I}(i_0^{(k)}=i)} \\
&= \prod_{i=1}^{s} {}_{(l)}p_i^{\sum_{k=1}^{N}\delta_l^{(k)}\bullet \mathrm{I}(i_0^{(k)}=i)} \\
&\equiv \text{Dirichlet}\left(1+\sum_{k=1}^{N}\delta_l^{(k)}\,\mathrm{I}(i_0^{(k)}=1),\ \ldots,\ 1+\sum_{k=1}^{N}\delta_l^{(k)}\,\mathrm{I}(i_0^{(k)}=s)\right)
\end{aligned} \qquad \mathbf{(6)}$$

$$\begin{aligned}
f\left(\underline{{}_{(l)}P_{i\bullet}} \mid \underline{{}_{(l)}P_{i\bullet}^{-}}, \underline{n}\right) &\propto \prod_{k=1}^{N}\left(\prod_{j=1}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}}\right)^{\delta_l^{(k)}} \\
&= \prod_{j=1}^{s} {}_{(l)}P_{ij}^{\sum_{k=1}^{N}\delta_l^{(k)} n_{ij}^{(k)}} \\
&\equiv \text{Dirichlet}\left(1+\sum_{k=1}^{N}\delta_l^{(k)} n_{i1}^{(k)},\ldots,1+\sum_{k=1}^{N}\delta_l^{(k)} n_{is}^{(k)}\right)
\end{aligned} \qquad \mathbf{(7)}$$

$$\begin{aligned}
f\left(\underline{\alpha} \mid \underline{\alpha^{-}}, \underline{n}\right) &\propto \prod_{l=1}^{m}\alpha_l^{\sum_{k=1}^{N}\delta_l^{(k)}} \\
&\equiv \text{Dirichlet}\left(1+\sum_{k=1}^{N}\delta_1^{(k)},\ldots,1+\sum_{k=1}^{N}\delta_m^{(k)}\right)
\end{aligned} \qquad \mathbf{(8)}$$

$$f(\delta^{(k)} \mid \delta^{(k)-}, \underline{n}) \propto \prod_{l=1}^{m} \left(\alpha_l \bullet {}_{(l)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}} \right)^{\delta_l^{(k)}}$$

$$\equiv \text{Mult}\left(1, \tfrac{1}{z}\left[\alpha_1 \bullet {}_{(1)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} {}_{(1)}P_{ij}^{n_{ij}^{(k)}}, \ldots, \alpha_m \bullet {}_{(m)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} {}_{(m)}P_{ij}^{n_{ij}^{(k)}} \right]\right) \quad \textbf{(9)}$$

$$\text{where } z = \sum_{l=1}^{m} \alpha_l \bullet {}_{(l)}p_{i_0^{(k)}} \prod_{i=1}^{s} \prod_{j=1}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}}$$

These distributions have a rather intuitive interpretation as well. The row updates are drawn from a distribution where the expected value is approximately the MLE for the row if the cluster assignments, δ, were known. The vector α is drawn from a distribution where the expected value is approximately the mixture proportions if, again, the cluster assignments were known. Lastly, the cluster assignments are drawn such that probability of each cluster is proportional to the mixture probability times the likelihood of the observation coming from the associated transition matrix.

2.4 Comments on the Gibbs sampling algorithm

The implementation of this algorithm initially fills in all of the transition matrices with s^{-1} and the vector α with m^{-1} and randomly assigns the δ to one of the m clusters. The algorithm first updates the initial state distribution, p, then each of the rows of P, then updates α, and lastly updates δ. This constitutes one iteration. After a large number of iterations (approximately 10,000, but this depends on the data and dimension of the problem) the sequence of parameter values will approximate the joint posterior distribution and hence, computation of arbitrary functionals of the posterior distribution is trivial.

As with all MCMC implementations of parameter estimation for mixture models, this method can suffer from the "label switching" problem. The posterior density for a particular labeling of the clusters is equal for any other permutation of the labels. If the clusters are "far apart" then it is *unlikely* that "label switching" would occur. However, with weak data or clusters that are very close "label switching" can be common place. In normal mixture models constraints are often imposed to insure identifiability. However, constraints alter the posterior distribution. A more appropriate method would detect switches in the labels and make corrections. Stephens (1996) has proposed such a method. I have made no effort to correct this problem and, therefore, rely on strongly informative data.

2.5 Simulation Experiment

I performed a two cluster/four state simulation experiment. I generated 4829 processes from P_1 and 171 processes from P_2.

$$P_1 = \begin{bmatrix} 0.26 & 0.43 & 0.13 & 0.18 \\ 0.06 & 0.37 & 0.19 & 0.38 \\ 0.86 & 0.05 & 0.04 & 0.05 \\ 0.32 & 0.38 & 0.20 & 0.10 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.07 & 0.17 & 0.19 & 0.57 \\ 0.12 & 0.15 & 0.08 & 0.65 \\ 0.35 & 0.03 & 0.32 & 0.30 \\ 0.27 & 0.17 & 0.14 & 0.42 \end{bmatrix}$$

The resulting posterior means (standard deviation) - 10,000 iterations, 9.7 minutes on a Pentium 200 MHz NT 4.0 machine.

$$P_1 = \begin{bmatrix} 0.26(0.003) & 0.43(0.004) & 0.13(0.003) & 0.18(0.003) \\ 0.06(0.002) & 0.37(0.003) & 0.19(0.003) & 0.38(0.004) \\ 0.86(0.004) & 0.05(0.002) & 0.05(0.002) & 0.05(0.002) \\ 0.32(0.004) & 0.38(0.004) & 0.20(0.004) & 0.10(0.003) \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.08(0.02) & 0.17(0.03) & 0.20(0.02) & 0.56(0.04) \\ 0.10(0.02) & 0.14(0.03) & 0.10(0.02) & 0.66(0.03) \\ 0.38(0.04) & 0.04(0.01) & 0.34(0.03) & 0.24(0.03) \\ 0.27(0.02) & 0.18(0.02) & 0.14(0.01) & 0.41(0.02) \end{bmatrix}$$

The true value of the mixture proportions, $\underline{\alpha}$, is (0.97,0.03). The posterior mean (standard deviation) of $\underline{\alpha}$ is
$\alpha_1 = 0.96$ (0.02)
$\alpha_2 = 0.04$ (0.02)

Process i is classified to the cluster to which it was most often assigned..

	P_1	P_2	
P_1	4814 (99.7%)	15 (0.3%)	4829
P_2	60 (35.1%)	111 (64.9%)	171

Table 1: Misclassification table

2.6 Comments on the simulation experiment

The small standard deviations indicate that using the posterior means as point estimates for $\underline{P}$ and $\underline{\alpha}$ is quite accurate. Process classification appears to have less accuracy. In this example, observations from P2, from which fewer observations were generated, are frequently misclassified (35%).

2.7 Scalability of Gibbs sampling

With relatively few iterations of the Gibbs sampler and a fairly small sample size the results were quite accurate. Preliminary experimentation indicates that this method may not be scalable for large state spaces. State spaces with 100 states and 10 clusters require about 3 seconds per iteration. Reasonable estimation therefore takes about 8 hours.

3 An EM approach to Markov process clustering

This section describes a parameter estimation approach based on the algorithm that maximizes a likelihood function that is slightly different than the one proposed in Section 2.1. EM algorithms iterate between obtaining maximum likelihood estimates for the unknown parameters given the complete data and computing the expected value of the missing data given the parameters. In this implementation, the algorithm iterates between computing maximum likelihood estimates for the transition matrices and reevaluating the cluster assignments.

3.1 Likelihood function

In the Gibbs sampling algorithm the $\underline{\delta}^{(k)}$'s were coerced to put probability 1 on one cluster and zero on all of the others. Then assessment of $\Pr(\ell \rightarrow k)$ comes directly from the distribution of the Monte Carlo sample of $\underline{\delta}^{(k)}$. As opposed to the Gibbs sampling algorithm the δ's now represent a probability vector where δ_ℓ indicates the probability that the process was generated from cluster ℓ. Despite this difference, strong similarities between the Gibbs sampling algorithm and the EM algorithm will be evident.

The likelihood function has to be modified to adapt to this alternate interpretation of δ. This version of the likelihood has the same meaning as (3) but its mathematical form would have been much more difficult to handle in the Bayesian framework.

$$\begin{aligned} f(\underline{n} \mid \underline{p}, \underline{P}, \underline{\delta}) &= \prod_{k=1}^{N} \Pr(\underline{n}^{(k)} \mid \underline{\delta}^{(k)}, \underline{p}, \underline{P}) \\ &= \prod_{k=1}^{N} \left[\sum_{l=1}^{m} \Pr(l \to k \mid \underline{\delta}^{(k)}, \underline{p}, \underline{P}) \bullet \Pr(\underline{n}^{(k)} \mid l \to k, \underline{\delta}^{(k)}, \underline{p}, \underline{P}) \right] \\ &= \prod_{k=1}^{N} \left[\sum_{l=1}^{m} \left(\delta_l^{(k)} \bullet {}_{(l)}p_{i_0^{(k)}} \prod_{i=0}^{s} \prod_{j=0}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}} \right) \right] \end{aligned} \qquad \textbf{(10)}$$

3.2 The algorithm

To initialize the algorithm I randomly assign the processes to the m clusters. That is, the δ's are randomly selected to represent assignment to one of the m clusters and α is the mean of the δ's. With this complete data, MLEs for the initial state distribution and the transition matrices are easy to calculate.

$${}_{(l)}p_i = \frac{\sum_{k=1}^{N} \delta_l^{(k)} \, \mathrm{I}(i_0^{(k)} = i)}{\sum_{k=1}^{N} \delta_l^{(k)}} \qquad \textbf{(11)}$$

$${}_{(l)}P_{ij} = \frac{\sum_{k=1}^{N} \delta_l^{(k)} n_{ij}^{(k)}}{\sum_{j=1}^{s} \sum_{k=1}^{N} \delta_l^{(k)} n_{ij}^{(k)}} \qquad \textbf{(12)}$$

This equation is similar to (7). Conditioning on the values of p and P, the cluster probabilities can be computed similar to (9).

$$\begin{aligned} \delta_l^{(k)} &= P(l \to k \mid \underline{n}^{(k)}, {}_{(l)}p, {}_{(l)}P) \\ &\propto P(\underline{n}^{(k)} \mid l \to k, {}_{(l)}p, {}_{(l)}P) P(l \to k) \\ &= \left({}_{(l)}p_{i_0^{(k)}} \prod_{i=0}^{s} \prod_{j=0}^{s} {}_{(l)}P_{ij}^{n_{ij}^{(k)}} \right) \bullet \alpha_l \end{aligned} \qquad \textbf{(13)}$$

Each vector $\delta^{(k)}$ is then normalized to sum to unity. Lastly, the mixture probability vector α is updated as the mean of the δ's.

3.3 A constrained EM algorithm

The EM algorithm is known to converge slowly in some situations. An alternative to algorithm proposed in section 3.2 is to force the δ's to assign probability one to one of the clusters and zero to the remaining. Hartigan's k-means algorithm is an example of this type of constrained EM algorithm for multivariate normal data. To make this modification, in lieu of equation (13), $\delta^{(k)}$ is assigned to the cluster from which has the highest probability of generating process k. The algorithm converges when an entire iteration is completed with no processes being reassigned.

3.4 Comments on the EM approach

A major drawback to the EM approach is the lack of standard errors. Gibbs sampling produces the estimates of the standard deviation of the marginal posterior density for any parameter of interest. EM, on the other hand, is solely a maximization method. Variants of the EM algorithm like the SEM algorithm (Supplemented EM) rely on normal approximations to the sampling distribution of the parameter estimates. In practice, these estimates are often quite reasonable. For the case at hand, however, the observed information matrix can be quite difficult to calculate. The "label switching problem" does not exist for EM algorithms.

3.5 A EM-Gibbs hybrid algorithm

The constrained EM algorithm lacks accuracy and detail but has the advantage of speed. The Gibbs sampler on the other hand can be used to compute arbitrary functionals of the distribution quite easily but takes several orders of magnitude longer to iterate to reasonable accuracy. Naturally a hybrid algorithm may be useful to borrow from the strengths and diminish the affect of the weaknesses of both algorithms.

In my implementation used for applied process cluster problems I iterate the constrained EM algorithm to convergence. The cluster assignments from the constrained EM algorithm provide initial assignments for the Gibbs sampler. Then, with a relatively short burn-in period, the Gibbs algorithm runs until it obtains decent estimates for the posterior means and variance of the parameters. Figure 1 shows the results of an example run on simulated data. The green line indicates the true value and the red lines indicate $\pm 2\sigma$.

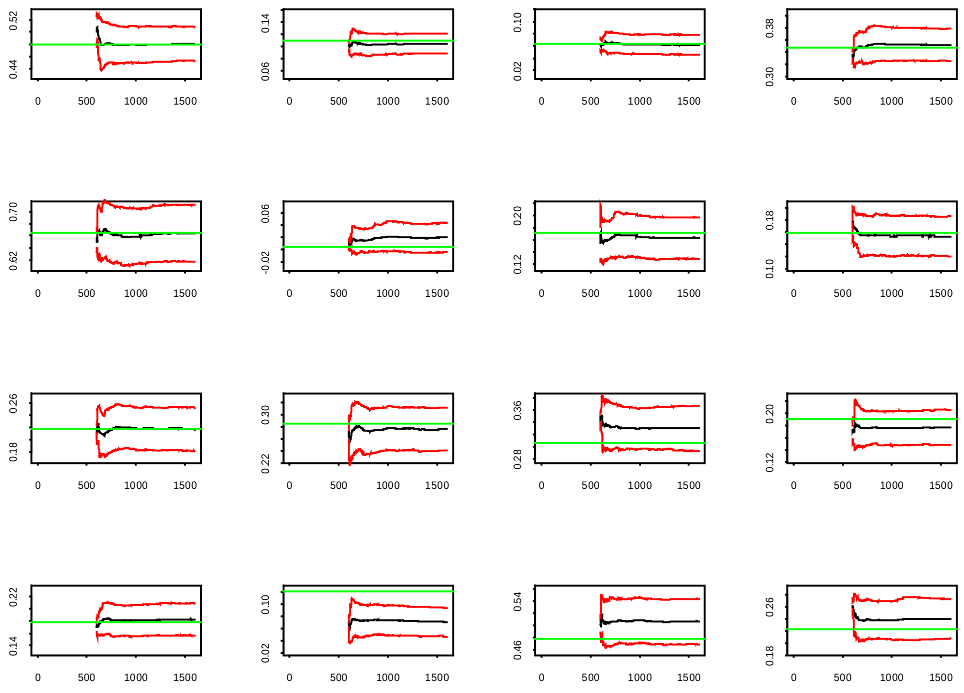

Figure 1: Estimate of P_1 from simulated data. Constrained-EM 4.0 seconds, Gibbs 3.0 minutes, 600 burn-in, 1000 draws, N=10,000, s=4, m=4.

Figure 1 shows that in a fairly short amount of time a large amount of information can be extracted from the data with a high degree of accuracy. As noted previously experiments with the larger state spaces required about 8 hours for 10,000 iterations. The first several thousand, however, were spent approaching the part of the distribution with most of the mass. With s=100 and m=10 the constrained EM algorithm tends to converge on this region of the distribution in about one minute. Quick and dirty point estimates are immediately available. The Gibbs sampler can then be run for a short amount of time (< 10,000 iterations) until confidence bands have reached a satisfactory level of accuracy.

4 Extensions

This section lists some useful extensions to the above algorithms

- Consider higher order Markov Chains using the linear model proposed by Raftery (1985) to constrain parameter explosion.
- Reversible jump MCMC to incorporate information contained in the data about the number of clusters.
- Include a post-processing step to check or fix problems arising from label switching.
- Make numerical optimizations so that this method could handle 100 states, 10 clusters, and several thousand observed sequences in less than 3 seconds per iteration.

5 Resources

1. Gelman, Carlin, Stern, and Rubin, *Bayesian Data Analysis*, Chapman & Hall, 1995.

2. Green, P. J. (1995). "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination". *Biometrika*, 82, 711-732.

3. Hastings, W. K. (1970). "Monte Carlo sampling methods using Markov chains and their applications." *Biometrika*, 57, 97-109.

4. Raftery, Adrian E. (1985). "A Model for high-order Markov Chains." Journal of the Royal Statistical Society B, 47, 528-539.

5. Ross, Sheldon M., *Probability Models 5th Edition*, Academic Press, 1993.

6. Stephens, Matthew (November, 1996). "Dealing with the Multimodal Distributions of Mixture Model Parameters." Available at http://www.stats.ox.ac.uk/~stephens/identify.ps.