Evolutionary MCMC Sampling and Optimization in Discrete Spaces

Malcolm J A Strens MJSTRENS@QINETIQ.COM
Future Systems Technology Division, QinetiQ, Cody Technology Park, Farnborough, Hampshire, GU14 0LX, United Kingdom.

Abstract

The links between genetic algorithms and population-based Markov Chain Monte Carlo (MCMC) methods are explored. Genetic algorithms (GAs) are well-known for their capability to optimize functions of discrete-valued variables, but the MCMC interpretation allows GA variants to be used for *sampling* discrete spaces (e.g. in Bayesian inference for machine learning). The GA crossover and mutation operators are modified to provide valid MCMC samples, and a new "exclusive-or" operator is introduced as an alternative way to recombine population members. This is shown to improve sampling performance in a medical diagnostic problem domain. The sampler can also be used within simulated annealing to provide a global optimizer that is similar to a GA in structure but has known convergence properties.

1. Introduction

Machine learning methods often generate sampling problems that must be solved efficiently. This is particularly true of Bayesian inference approaches to 'explaining' observed data using compact models. Examples of such models are decision trees, Bayesian networks, and sets of logic statements. In this paper the focus is on the discrete-valued parts of these models, rather than any continuous parameters that must also be inferred. The discrete-valued parts might be structural information about the architecture of the model (dependency arcs in a Bayesian net, for example) or assignments of boolean-valued states in the model. In different domains this information might be called the *structure*, *hidden states* or the *chromosome*. Throughout this paper, a medical diagnosis problem is used as an example: the hidden state defines the presence or absence of a set of diseases in the patient.

The requirement is often not just to find the the maximum likelihood structure, but instead to assign a probability to every possible structure that could explain a set of observations. This provides a much richer source of information for subsequent stages of processing (e.g. decision-making or data fusion). When the space of structures is very large or infinite, it becomes necessary to *sample* rather than enumerate this space.

1.1. Sampling for Bayesian Inference

Let B be the set of unique bit-strings that describe possible structures. For simplicity, assume that their lengths are all equal to some constant m. Therefore the set can be indexed using a natural number $1 \leq i \leq 2^m$. In Bayesian inference, a model $P(D|b)$ is assumed for observations D given structure $b \in B$. The observations in the medical application are the *findings* of a series of tests, indicating the presence or absence of disease symptoms. There is also a prior distribution $P(b)$ for the likelihood of each structure: this assigns low probabilities to rare diseases, for example. Then the *posterior* distribution on b is given by Bayes rule:

$$P(b|D) = \frac{P(D|b)P(b)}{P(D))}$$

If the observations (D) are given, this becomes a *target* function $\pi_D(b)$. The *maximum a posteriori* solution for b is then given by $\hat{b} \equiv \arg\max_b \pi_D(b)$. This solution can be found by discrete optimization techniques such as genetic algorithms, random search, tree searches, etc. However, it is often the case that $\hat{b}$ is of little use for decision-making or subsequent processing: the most-likely cause of the observations is not necessarily the correct one.

Suppose that the Bayesian inference task is to interpret observations in a *decision-making* system. The decision u will lead to a set of outcomes or payoffs given by a function $R(b, u)$. Then the optimal decision is given by:

$$\hat{u} \equiv \arg\max_u \sum_b \pi(b)R(b,u) \equiv E_{\pi(b)}[R(b,u)]$$

If the expectation cannot be obtained by exhaustive evaluation of $\pi(b)$ over $b \in B$, then an estimate can be obtained by *sampling* N independent hypotheses $H \equiv \{b^{(1)}, \ldots b^{(N)}\}$ from $\pi(b)$ and evaluating the average:

$$\sum_{b^{(j)} \in H} R(b^{(j)}, u)/N$$

Alternatively, assume that Bayesian inference is to be used to provide information from a single source of data (e.g. a sensor) that will later be fused with other sources, before attempting to infer the hidden state. Again, the optimal solution can be approximated using sets of samples. If each source k yields a sample set H_k of size N, an estimate for the probability of some hidden state b is given by:

$$\prod_k \frac{count(k,b)}{N}$$

where $count(k,b)$ is the number of occurrences of b within H_k.

2. Genetic Algorithms

Genetic algorithms (Holland, 1975) are introduced here to identify a source of proposal mechanisms that can be used in the sampling algorithm. The GA searches through the set B of fixed-size binary strings ("chromosomes"), to maximize some "fitness" function[1] $f(b)$. At any given time, the state of the GA is a population of chromosomes $(b_1, \ldots, b_N)$.The population is modified by creating new proposals in batches. The next generation (also of size N) is then generated from the new proposals (and the existing population) according to a selection mechanism which prefers fitter individuals. The genetic algorithm may be run for a fixed number of generations, or until some condition on the average population fitness (or its rate of change) is met.

Each proposal is obtained by selecting two parent chromosomes and applying *crossover* to create a new chromosome. An example is standard 2-point crossover:

$$b'_l = \begin{cases} b_{i,l}, & \{(l-n+L) mod L\} < s; \\ b_{j,l}, & \text{otherwise.} \end{cases}$$

[1]For maximum likelihood estimation, the fitness function might be $f(b) \equiv exp(-E(b))$ where $E(b)$ is a measure of the prediction error on observed data, given model parameters b.

where b_i and b_j are the parent chromosomes and the offset n and length s are chosen uniformly from $\{1, \ldots, m\}$. Each proposal is then subject to mutation:

$$b_l \leftarrow \begin{cases} 1 - b_l, & \text{with probability } \rho; \\ b_l, & \text{otherwise.} \end{cases}$$

where ρ is some small constant mutation probability. While crossover allows fit substrings to grow in frequency within the population, mutation enables the population to move out of local minima.

This paper focusses on fixed size chromosomes of binary digits, but the broader field of evolutionary algorithms addresses problems in which the chromosome is a vector of real numbers[2], or has variable size (e.g. genetic programming).

Evolutionary algorithms can benefit from estimating the structure of the current population, in order to obtain new proposals that are predicted to have a high fitness. Population-based incremental learning (PBIL) (Baluja & Caruana, 1995) uses an adaptive probability vector (learnt on-line from the population) to control the likelihood of selecting 0 or 1 in each string position. Higher-order estimation methods, for example the Bayesian network used in the *Bayesian Optimization Algorithm* (BOA), can capture multi-dimensional structure in the existing population (Pelikan et al., 1999). We will adapt proposals to the high-order structure of the current population by using the *difference* between two members of the population, combined with a third, to obtain a proposal[3]. Therefore our approach may have the benefits of these "estimation of distribution" approaches while avoiding the process of parametric model-fitting. It can operate with arbitrarily complex (e.g. multi-modal) populations whereas estimation-based approaches will be limited by the chosen class of models.

3. Population-based MCMC Sampling

Markov Chain Monte Carlo (MCMC) is a means of sampling hypotheses from some target density $\pi(b)$ that is known up to some normalizing constant[4] Z. MCMC can be interpreted as a form of importance sampling in which the proposal distribution *depends on* the current state of the chain (e.g. by making a

[2]e.g. differential evolution; Storn and Price (1995)

[3]Strictly, the difference between the first two population members captures the first-order of *relational* structure; the third population member accounts for absolute high-order structure because it is a sample from the population

[4]For Bayesian inference problems this means that $P(D)$ need *not* be known.

random change). Each proposal is accepted or rejected according to the usual importance sampling rule. Several MCMC chains can be run in parallel, to obtain evolutionary or "population-based" methods that appear similar in structure to a genetic algorithm but perform sampling rather than optimization.

3.1. Markov Chain Monte Carlo

MCMC works by constructing a Markov chain over B that has π as its invariant distribution. A Markov chain on B is defined by a transition matrix T of size $|B|\times|B|$ which determines the time-evolution of a state variable X_t:

$$T_{ij} \equiv P(X_{t+1} = b_j | X_t = b_i)$$

A necessary condition is that the vector of probabilities Π with elements $\Pi_i = \pi(b_i)$ be a fixed point of the transition function ($\Pi = T\Pi$). An appropriate choice is given by:

$$T_{ij} \equiv \begin{cases} 1, & (\pi(b_j) > \pi(b_i)); \\ \frac{\pi(b_j)}{\pi(b_i)}, & \text{otherwise.} \end{cases}$$

Another requirement is ergodicity of the Markov chain: there must be a finite path with non-zero transition probability from every state to every other state. Sampling from π is achieved by simulating the Markov chain, and outputting its states at irregular intervals of mean duration τ. If each interval is sufficiently large, successive states will be independent. Independence is not a requirement for evaluating expectations such as the decision evaluation $E_\pi[R(b,u)]$, but chains which *mix* (achieve independence) more rapidly require shorter sequences to provide accurate estimates.

To design an effective MCMC sampler requires an appropriate *proposal* mechanism. The proposal mechanism, together with an acceptance rule define the transition function of the Markov chain. Specifically, a proposal distribution $P(b'|b)$ is used to generate a proposal b' for the next state of the chain, given the current state is $X_t = b$. The proposal distribution is symmetric if $P(b'|b) = P(b|b')$ for all (b, b'). An *acceptance rule* decides whether the next state of the chain (X_{t+1}) will be b' or b. The Metropolis acceptance rule (Metropolis et al., 1953) is given by:

$$P(X_{t+1} = b') = \begin{cases} 1, & (\pi(b') > \pi(b)); \\ \frac{\pi(b')}{\pi(b)}, & \text{otherwise.} \end{cases}$$

For a symmetric proposal distribution, the Metropolis acceptance rule ensures that the transition matrix has Π as a fixed point. The more general Metropolis-Hastings rule can be used to account for non-symmetric proposal distributions. Most MCMC methods (including those that use the Metropolis-Hastings rule) have a transition function that satisfies the *reversibility* property (also called "detailed balance"): $T_{ij}\Pi_i = T_{ji}\Pi_j$.

3.2. The Mutation Proposal

A simple *mutation* proposal is sufficient to meet the above requirements. Mutation constructs b' by randomly inverting each bit in b according to some probability ρ. Therefore:

$$P(b'|b) = \rho^{\Delta(b,b')}(1-\rho)^{(m-\Delta(b,b'))}$$

where $\Delta(b, b')$ is the number of bits that differ between the strings b and b' and m is the size of the strings. Symmetry of the mutation proposal density follows from observing that $\Delta(b, b') = \Delta(b', b)$ for all (b, b'). The resulting chain is ergodic because $P(b', b) \geq \rho^m$ for all (b', b).

The key to effective sampling performance is to design the proposal distribution to minimize mixing time. Where possible, prior knowledge about the problem should be incorporated into the proposals. For example, if b represents the structure of a decision tree, natural proposals would be common tree operations such as splits, merges and rotations.

3.3. Evolutionary Monte Carlo

It is possible to run multiple MCMC sampling chains in parallel. One benefit is that each chain can be started with a different (random) state, and so reasonable samples may be obtained even when the chains have not "mixed". However, even when running multiple chains, a "burn-in" time must be allowed for each chain to move from its random starting state to a high-probability region of the target distribution. Therefore, for any given problem there is some (finite) optimal number of chains N. However, there is another potential advantage of running multiple MCMC sampling chains: *interaction.* At any one time, the set of chains provide a *population* that is spread across different regions of the target distribution. It is reasonable to expect that the population as a whole contains useful information about the direction in which any particular population member could explore to find regions of higher probability. That is, the proposal distribution for one population member can exploit the information contained in the others. The evolutionary Monte Carlo (EMC) method was introduced recently (Liang & Wong, 2001). Much previous work in the evolutionary computation field proposed algorithms with stochastic acceptance of proposals that could be used for robust optimization, but were not valid MCMC sampling algorithms (Lozano et al., 1999; Mahfoud & Goldberg, 1995).

The algorithm described here differs significantly from EMC. We do not run each sampling chain at a different temperature (parallel tempering; see Geyer (1992)). Strens et al. (2002) showed that a suitable proposal operator capable of adapting to the shape of the population enabled very effective sampling in real spaces at a single temperature. This "differential evolution sampler" (DES) simplifies the implementation greatly, and allow outputs to be taken from any one of the sampling chains. It also provides a closer resemblance to conventional evolutionary algorithms because all the individuals have the same fitness criterion. Here, we attempt an analogous approach in the discrete-state case, where we must find a similarly adaptive proposal operator that acts on bit-strings.

When using a population it is again important to have a proposal distribution of a form that ensures valid sampling. Let $\beta \equiv (b_1, \ldots, b_N)$ be the joint state of the whole population, drawn from the product-space B^N. We construct a Markov chain on this space that has as its invariant distribution:

$$\phi(\beta) = \pi(b_1)\pi(b_2)\ldots\pi(b_N)$$

Sampling from π is achieved by sampling β from ϕ, then returning any one of the members $\{b_i : 1 \leq i \leq N\}$.

Proposals in this new Markov chain are defined in terms of the joint population state β. However, consider firstly a proposal that would alter only one member (b_i) of the population (with i chosen at random). A transition matrix T satisfying the invariance and ergodicity requirements for sampling from π also ensures the population's invariant distribution is ϕ. Therefore any combination of proposal distribution and acceptance rule that is valid for a single chain sampler can be applied to a randomly chosen member of the population. Furthermore, in constructing such a proposal the states of all other population members $\{b_j : j \neq i\}$ can be exploited.

3.4. A New Proposal

We now introduce a new proposal of this kind:

$$b' = b_i \otimes (b_j \otimes b_k)$$

where i, j, k are mutually unique population members. The $\otimes$ symbol indicates bit-wise exclusive or:

$$(b \otimes b')_l = \begin{cases} 1, & b_l \neq b'_l; \\ 0, & b_l = b'_l. \end{cases}$$

where l indexes the individual bits of the strings b and b'. The proposal distribution is symmetrical because $b' \otimes (b_j \otimes b_k)$ regenerates b_i.

Why should this "xor" proposal be useful? A similar proposal mechanism was introduced recently for sampling in continuous state spaces (Strens et al., 2002). In that case the proposal was of the form:

$$x' = x_i + F(x_j - x_k)$$

where x_i, x_j, x_k are real-valued state vectors and F is a scalar constant. With $F = 1$ the effect is to generate a proposal x' that differs from its parent x in the same way that x_j differs from x_k (in terms of vector arithmetic). This type of proposal allows the population to explore non-isotropic structures such as ridges in the target function, because the vector differences are typically aligned with the direction of the ridge. The aim here is the same, but to obtain a proposal mechanism that is applicable to binary vectors; therefore the "+" and "-" are replaced by $\otimes$. Observe that:

$$\{b' = b_i \otimes (b_j \otimes b_k)\} \Rightarrow \{(b' \otimes b) = (b_j \otimes b_k)\}$$

Reading $(A \otimes B)$ as "the difference between A and B" the effect is to generate a proposal b' that differs from its parent b in the same way that b_j differs from b_k. We expect this to provide a proposal density that adapts to typical variability of the population: an experimental evaluation will be performed to determine whether this is true.

3.5. The Crossover Proposal

Unlike the mutation and xor proposals introduced above, the crossover operation that is often used in genetic algorithms must be adapted carefully to meet the requirements for population-based MCMC proposals. Crossover works by selecting two parents b_i and b_j from the population and constructing a child that consists of some genetic material from each. For example:

$$b'_l = \text{choose}(b_{i,l}, b_{j,l})$$

where "choose" selects randomly between its two inputs according to some fixed probability. If the proposal is accepted then b_i, b_j or some other member of the population is replaced (overwritten) by b'.To ensure reversibility of the population chain, it is necessary for the proposal to select two parent strings and replace them *both* with child strings, preserving every bit of each parent within one of the children. Therefore if child 2 receives bit $b_{i,l}$, child 1 must receive $b_{j,l}$. Suppose that a crossover proposal generates children (c_1, c_2) from parents (b_i, b_j). Then the Metropolis rule on the joint population state has acceptance probability:

$$\begin{cases} 1, & \pi(c_1)\pi(c_2) \geq \pi(b_i)\pi(b_j); \\ \frac{\pi(c_1)\pi(c_2)}{\pi(b_i)\pi(b_j)}, & \text{otherwise.} \end{cases}$$

If the proposal is accepted, both parents are replaced; otherwise there is no change to the population. The chain is reversible because a proposal that regenerates the parents is equally likely to be chosen at the next step. Note that the crossover proposal requires two evaluations of the target distribution π (at c_1 and c_2; the values at b_i and b_j will already be stored). This additional computational cost is taken into account in determining whether crossover can be beneficial in a sampling context. An interesting property of this form of crossover is that there is a special form of π where $\pi(c_1)\pi(c_2) = \pi(b_i)\pi(b_j)$ always. This is true if the bits of b are independent in terms of their influence on π. (i.e. $\pi(b)$ is of the form $\prod_l f_l(b_l)$.) In this case, crossover proposals will always be accepted because they cannot increase or decrease the joint probability of the population, $\phi(\beta)$.

4. The Sampling Algorithm

The 3 types of proposal (mutation, crossover and xor) can be used within a population-based MCMC method, and can together meet the requirements for valid sampling. Given an initial population β of random bit-strings, and the value of π for each, proposals are repeatedly accepted/rejected until the budget of function evaluations is exhausted. The proposal type is selected according to the probabilities (p_M, p_C and p_X), then the following steps are applied:

Mutation

1. Select a random parent b_i and apply mutation to generate proposal b'.

2. Accept or reject b' using the Metropolis rule, replacing b_i accordingly. (Consumes 1 target distribution evaluation.)

Crossover

1. Select two mutually exclusive parents (b_i, b_j) uniformly and randomly. Generate two children c_i, c_j by choosing to swap each bit with probability 1/2.

2. Accept or reject both children *together* using the Metropolis rule, replacing (b_i, b_j) accordingly. (Consumes 2 target distribution evaluations.)

Table 1. Choices of proposal probabilities for experimental evaluation.

Experiment name	p_M	p_C	p_X
MUT	1	0	0
MUT + CRX	2/3	1/3	0
MUT + XOR	1/2	0	1/2

Xor proposal

1. Select three mutually exclusive parents (b_i, b_j and b_k) uniformly and randomly. Generate $b' = b_i \otimes (b_j \otimes b_k)$.

2. Accept or reject b' using the Metropolis rule, replacing b_i accordingly. (Consumes 1 target distribution evaluation.)

To ensure that the Markov chain is ergodic requires that $p_M > 0$. Otherwise the only constraint is $p_M + p_C + p_X = 1$. Table 1 shows the choices used in the experimental evaluation. These are chosen to balance the budget of evaluations of π equally between the selected proposal types.

A random member of the population can be provided as output on every time step. Although these outputs are highly correlated from one step to the next, the resulting expectation (or any other statistics computed from the samples) will be asymptotically unbiased.

5. Evaluation

In order to demonstrate the effectiveness of the new sampling algorithm, we introduce an application problem from medical diagnostics that offers a difficult discrete sampling problem. A comparison of sampling performance for different proposal combinations is then presented. The effect of population size is also evaluated.

5.1. The QMR-DT architecture

Consider the problem of diagnosing which diseases are present in a patient, given a set of measurements ("findings"). Each patient may have zero, one or many diseases. The QMR-DT network architecture (Figure 1) specifies a statistical model for the relationship between diseases and findings, that can be calibrated using recorded data (Shwe et al., 1991; Jaakkola & Jordan, 1999). The model is of the form:

$$P(f_i = 1|b) = 1 - (1 - q_{i0}) \prod_l (1 - q_{il})^{b_l}$$

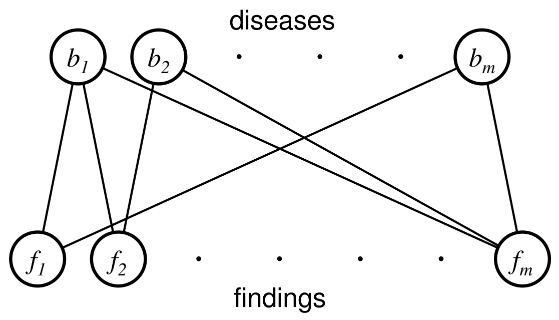

Figure 1. The QMR-DT architecture.

f_i is the binary value of finding i. q_{i0} is the *leak probability* for that finding. If q_{i0} is close to 1 then the finding is common even when no disease is present. q_{il} is the association between disease l and finding i. If q_{il} is large then the finding i often results from disease l; if $q_{il} = 0$ disease l does not influence finding i and the corresponding arc can be omitted from the network diagram.

In order to perform inference, a prior $P(b)$ must also be specified. The simplest approach is to assume that the diseases are independent:

$$P(b) = \prod_l p_l^{b_l}(1 - p_l)^{(1-b_l)}$$

where p_l is the prior probability for disease l. Random instances of this architecture were generated according to:

1. Number of diseases $m = 20$
2. Number of findings $n = 80$
3. Disease prior $p_l \sim U[0, 0.5]$
4. Leak probability $q_{i0} \sim U[0, 1]$
5. Association between disease l and finding i:

$$q_{il} \sim \begin{cases} 0, & \text{with probability 0.9;} \\ U[0,1], & \text{otherwise.} \end{cases}$$

40 such instances were generated, and for each a sampling test problem was created as follows. Firstly, a plausible hidden disease state b_{truth} was created by sampling from the disease prior. Then a set of findings were sampled according to $P(f|b)$ for that instance. The sampling problem is to obtain the distribution of possible diseases given the findings; i.e. to perform a diagnosis. Therefore the target distribution is $\pi(b) \equiv P(b|f)$. For any give proposal b it is possible to evaluate $\pi(b)$ up to an unknown constant (equal to $P(f)$) because $P(b|f) = P(b, f)/P(f)$ and $P(b, f)$ is easily computed in the architecture.

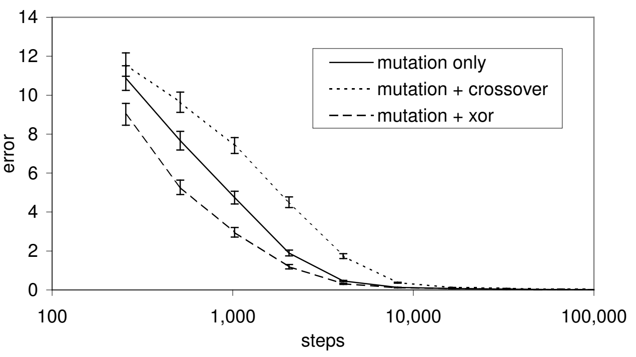

Figure 2. Sampling performance for differing proposal combinations.

5.2. Measuring Sampling Accuracy

Samples are generated by applying the algorithm described in section 4, and outputting a random population member after every step. (One step corresponds to one evaluation of the target distribution.) For each disease l the empirical proportion of samples for which $b_l = 1$ is computed by exhaustive evaluation. This gives a vector of probabilities[5] ψ. The true value of $\mu_l \equiv P(b_l = 1|f)$ was also computed by exhaustive evaluation of the 2^m disease combinations. In an effective sampler, we should find that $\psi \rightarrow \mu$. A measure of the difference between these vectors is given by:

$$\sum_l (\mu_l - \psi_l)(\log_2(\mu_l) - \log_2(\psi_l))$$

This symmetrical measure is zero when $\mu = \psi$ and positive otherwise. This measure is much more informative than the full KL-divergence (between the true probability distribution of diseases and the set of samples) because the latter is infinite if *any* state that has non-zero probability is not present in the set of samples.

Figure 2 shows the sampling accuracy over 100,000 steps for each of the proposal combinations in Table 1. The *standard errors* indicated by error bars were computed from the set of performance measures obtained over the 40 random problem instances. There are significant differences in performance between the three experiments. The crossover operator makes performance worse rather than better, which suggests that the extra cost (two target distribution evaluations) outweighs any benefit. In contrast, the xor proposal mechanism shows a significant improvement (reducing error by 38% at 1024 steps). In this evaluation we have limited the size of the state space to

[5]NB It is not the case that $\|\psi\| = 1$ because the patient may not have exactly one disease.

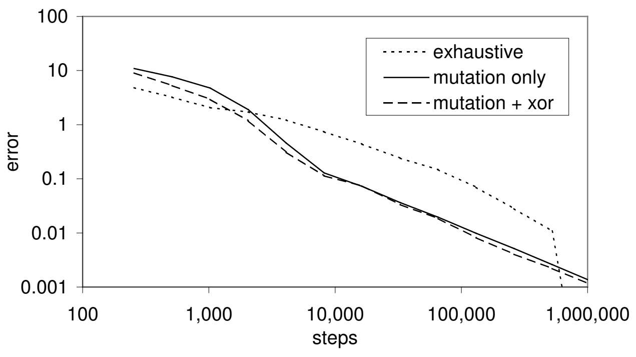

Figure 3. Long-term performance compared with exhaustive evaluation.

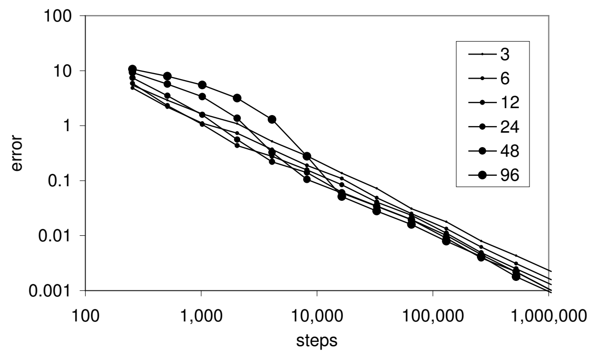

Figure 4. Sampling performance for differing population sizes.

2^{20} in order to be able to evaluate it exhaustively (to validate performance). Greater performance improvements might be possible on larger and more complex spaces in which the "local search" performed by mutation is inadequate. (Conversely, the rejection rate within the Metropolis rule for the xor-proposal is likely to rise as dimensionality grows, potentially reducing performance.)

Figure 3 shows sampling performance over a much longer period (log-log scale). We observe that performance continues to improve at the same rate. Given enough time, the performance of the mutation mechanism approaches that of the xor-operator. This suggests that both populations have become adequately spread across the high probability regions of the target density and are essentially in equilibrium. The "exhaustive" performance line indicates the result of systematically visiting every state (in a random order). This verifies that the problem is non-trivial and that the sampling approaches offers some benefit over a "brute-force" method. The error associated with this method only becomes zero after 2^{20} (about a million) steps. Exhaustive evaluation rapidly becomes infeasible as the dimensionality of the state space increases.

Figure 4 shows the effect of population size on performance of the xor-proposal sampler. Population size is important until about 10000 steps, with a moderate population size (12) giving the best performance. This probably offers the best trade-off between diversity (number of individuals) and the rate at which each state in the chain is being updated. After this initial "exploration" phase, the performance of all the population sizes becomes similar, although larger populations become slightly better. This is probably because the larger populations provide more consistent coverage of the whole target distribution.

6. Simulated Annealing Variant

We have taken proposal operators that would normally be used for evolutionary optimization and applied them to accelerate sampling. However simulated annealing (SA) (Neal, 1998) is a means of using a sampler to perform global optimization. It works by sampling from π^T/Z_T while gradually reducing the temperature parameter T. (Z_T is the normalizing constant $\sum_b \pi^T(b)$.) As T decreases towards zero, the distribution becomes "sharper"; i.e. probability mass moves towards the global maximum (or maxima) of π. Therefore simulated annealing allows us to obtain a global optimizer from the new sampling method. This is particularly useful in difficult integer programming or SAT (satisfiability) problems where the aim may be to find *any* solution rather than *every* solution.

For example, we tested the SA variant of the sampling algorithm on the "8-queens" problem. The aim is to place 8 queen chess pieces on an 8-by-8 board so that none share a common rank, file of diagonal. By encoding the position of each in its file as a 3-digit binary number, a 24-bit optimization problem was obtained. The target density was $\pi(b) \propto exp(-hits(b)/T)$ where $hits(b)$ was the number of constraints broken in board configuration b. 2^{16} sampling steps were sufficient for the SA algorithm to find a solution (with probability > 0.99). This was using a population size of 24, and a uniform linear reduction in temperature from 1 to 0. (All three proposal options gave this level of performance.)

7. Conclusions and Future Work

We have developed an evolutionary Monte Carlo algorithm for sampling in discrete spaces. The population is not only used as a means to obtain independent sampling chains: it is exploited by a proposal mechanism that uses relational information (between existing population members) to move about the state space more effectively. Our experimental evaluation showed a significant improvement in sampling performance for a representative machine learning problem. The algorithm has a simple implementation compared with existing methods for exploiting population structure ("estimation of distribution" approaches such as BOA); but a comparative evaluation should be performed.

We also showed that simulated annealing can make use of the sampler to obtain a global optimizer. This has an advantage over conventional genetic algorithms in that the rate of cooling can be used to control the trade-off between the risk of becoming trapped in a local minimum and the total computational cost.

The capability to sample from discrete spaces has wide applicability in machine learning, particularly in the convergence between statistical and symbolic learning. Often it will be necessary to be able to work with probability distributions over strings of symbols and graphical model structures. The method described here works solely with binary strings of fixed length, complementing previous work that focussed on real-valued states (Strens et al., 2002), but it could be naturally extended to work with mixed real/discrete state-spaces within the EMC framework, and to spaces of variable dimension using reversible jump MCMC methods (Green, 1995).

The general purpose proposal mechanisms we have described may work well for the test problems presented, but better performance is likely to be achieved by making use of domain knowledge. If the problem is to find a graphical structure that best explains some data, standard operations for working with that structure (e.g. node and arc deletion/insertion in a graph) can be modified to act as suitable proposal mechanisms.

Acknowledgements

This research was funded by the UK Ministry of Defence Corporate Research Program.

References

Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. *Proceedings of the Twelth International Conference on Machine Learning* (pp. 38–46). San Mateo, CA: Morgan Kaufmann Publishers.

Geyer, C. (1992). Practical Markov chain Monte Carlo (with discussion). *Statistical Science*, *7*, 473–511.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, *82*, 711–732.

Holland, J. H. (1975). *Adaptation in natural and artifical systems*. MI: University of Michigan Press.

Jaakkola, T., & Jordan, M. I. (1999). Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, *10*, 291–322.

Liang, F., & Wong, W. H. (2001). Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association*, *96*, 653.

Lozano, J. A., Larrañaga, P., Graña, M., & Albizuri, F. X. (1999). Genetic algorithms: bridging the convergence gap. *Theoretical Computer Science*, *229*, 11–22.

Mahfoud, S. W., & Goldberg, D. E. (1995). Parallel recombinative simulated annealing: a genetic algorithm. *Parallel Computing*, *21*, 1–28.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Neal, R. (1998). *Annealed importance sampling* (Technical Report 9805 (revised)). Department of Statistics, University of Toronto.

Pelikan, M., Goldberg, D. E., & Cantu-Paz, E. (1999). BOA: The bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 525–532). Orlando, Florida, USA: Morgan Kaufmann.

Shwe, M., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., & Cooper, G. (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: Part I. The probabilistic model and inference algorithms. *SIAM Journal on Computing*, *30*, 241–250.

Storn, R., & Price, K. (1995). *Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces* (Technical Report TR-95-012). Berkeley, CA.

Strens, M., Bernhardt, M., & Everett, N. (2002). Markov chain monte carlo sampling using direct search optimization. *Proceedings of the Nineteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

© QinetiQ Ltd 2003