Hidden Markov Models applied to Data Mining

Andrea Marin

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino 155, 30172 Venezia Mestre, Italy
marin@dsi.unive.it

Abstract. Final task for the course *Data Mining*, BISS 2006, prof.sa Rosa Meo.

1 Stochastic Finite State Automata (SFSA)

In this section we analyse the Hidden Markov Models (HMM) as part of a larger theory, the automata theory, as suggested in [2]. This allows us to show on one hand the relations between these models and others which are well-known pointing out similar and different aspects, on the other to point intuitively out the main advantages and the application fields of HMM.

Before introducing formally Stochastic Finite State Automata we consider useful, especially for the clearness of the notation, recalling the *classic* definition of finite state automaton as given in [6].

Definition 1 (Finite State Automata) *A Finite State Automaton (FSA)* $\mathcal{M}$ *is an abstract machine consisting of:*

- *A set of states* $\mathcal{Q} = \{I, 1, \ldots, k, \ldots, K, F\}$ *including the initial state* I *and the final state* F *(accepting state)*[1]
- *A set* $\mathcal{Y}$ *of input symbols.*
- *A set* $\mathcal{Z}$ *of output symbols.*
- *A state transition function* f*. If* q_t *is the state of the automaton* M *at time* t *then* $q_t = f(y_t, q_{t-1})$*.*
- *An emission function* g*. In the* Mealy *automata the emission function depends on the transition between states, i.e. if* z_t *is the automaton output at time* t*,* $z_t = g(q_t, q_{t-1})$*; In the Moore automata the emission function depends only on the current automaton state, i.e.* $z_t = g(q_t)$*. Clearly we know that the Mealy automata class is equivalent to the Moore automata one.*

We just add some notes on the given definition. In the automata theory it is usually introduced a more accurate definition than the one we have just given, for example by the definition of the input and output alphabet. In this paper, considering the applications we are going to explore, we try to pay our attention

[1] The definition can be extended considering more initial states and/or more final states.

just on the main aspects. For example in the *speech recognition* applications, the input and the output symbols are vectors with real number components which are the result of the spectral analysis of an acoustic signal at a given time, while in the *pattern recognition* applications we still have a discrete input and output alphabet.

Another important note, is that the determinism of the automata is given by the determinism of the transition function f and the emission function g. When the emission function or the transition function are probabilistic, then we are treating a *stochastic* automaton.

Markov Models (MM) In [2] the authors suggests that the classic Markov Models can be seen as SFSA with a probabilistic transition function and a deterministic emission function (the identity) depending on the current state. The probability of observing the automaton in the state q at time t depends only on the state of the model at time $t-1$. This feature of the transition law means that the MM are memoryless[2]. We can consider the state of the chain as the emission function.

With MM we can introduce and give the solution of two interesting problems:

1. Given a sequence consisting of T states $X = x_1 \dots x_T$ and a Markov Model $\mathcal{M}$, which is the probability that, starting from the initial state I, $\mathcal{M}$ gives X as sequence of output symbols and then it terminates[3]?
2. Let us consider the Markov Model $\mathcal{M}$ with K states. We can represent the transition probability between states as a $K \times K$ dimension matrix P whose element p_{ij} represents the probability of observing the state j just after i. Formally if λ represents the parameters associated to $\mathcal{M}$, i.e. the probability transition matrix, then the problem consists in estimating, given one or more training sequences as training set, $\lambda^* = \arg\max_\lambda p(X|\mathcal{M}, \lambda)$.

Both these problems have an easy solution. In the first case, noting that the requested sequence is the exact sequence of states of the Markov Model, we have that the probability of observing $X = x_1 x_2 \dots x_T$ is:

$$p(X) = p(F|x_T)p(x_1|I)\prod_{t=2}^{T} p(x_t|x_{t-1})$$

To solve the second problem we have to find the estimation of the parameter λ such that the probability of observing the training set is maximum. The solutions usually achieve the result by optimising a *maximum likelihood criterion*, which for the memoryless property is easily:

$$P(x_t = l|x_{t-1} = k) = \frac{n_{kl}}{n_k},$$

[2] This restriction can be relaxed by the introduction of Markovian Models of order t. In this paper we don't treat them because it is always possible to associate to a MM of order t a simple MM

[3] The idea of termination is strictly related to the choice of the introduction of the MMs based on automata theory.

where n_{kl} represents the number of times which the state l follows the state k in the training set, while n_k represents the number of times which the state k is visited. Thus we can use $P(x_t = l|x_{t-1} = k)$ as estimation for λ^*.

Hidden Markov Models (HMM) From the automata theory point of view, a Hidden Markov Model differs from a Markov Model for two features:

1. It is not possible to observe the state of the model, i.e. q_t is not given;
2. The emission function is probabilistic.

We have to think that somehow there are two dependent stochastic processes, for the emission symbols we know the law which regulates the process and we can observe the process itself, while the other process is not observable and all we know is the law which regulates it.

So an Hidden Markov Model has all the parameters of a Markov Model with the addition of the stochastic emission function, which means the probability that, given the state q_t, the symbol x_t is observed.

For the HMM we can point out three problems which are extremely important and interesting ([10]):

1. Given the sequence $X = x_1 \dots x_T$ which is the probability to observe it on the model $\mathcal{M}$?
2. Given the sequence $X = x_1 \dots x_T$, which is the *optimal*[4] sequence of states of $\mathcal{M}$ which produces X?
3. given a training set X, how can we estimate the parameters of the model $\mathcal{M}$ to maximise $P(X|\mathcal{M})$?

Now the solutions of the problems are not trivial. In the following sections we analyse how to efficiently solve these problems. Before going on, we formally give a definition for HMM which is no more strictly related to the automata theory, which is the classical definition introduced in the famous Rabiner tutorial [10].

Definition 2 (Hidden Markov Model) *An Hidden Markov Model consists of:*

- *N, the number of states in the model; we denote each state with S_i where $1 \leq i \leq N$ and let S be the set of the states;*
- *M, the number of distinct observation symbols. We denote individual symbols as $V = \{v_1, \dots, v_M\}$;*
- *A, the state transition probability matrix, which is a $N \times N$ matrix; we denote with q_t the state of the model at time t, so $a_{ij} = P(q_{t+1} = S_j|q_t = S_i)$ with $1 \leq i, j \leq N$;*
- *B, the observation symbol probability distribution for each state: $B = \{b_j(k),\ 1 \leq j \leq N, 1 \leq k \leq M\}$, where $b_j(k)$ is the probability to observe v_k at a certain time t when the state of the model at the same time is S_j;*

[4] Later we explore better the meaning of this request

– π, *the initial state distributions* $\pi = \{\pi_i,\ 1 \leq i \leq N\}$ *where* π_i *is the probability that the initial state of the model is* S_i[5].

$\mathcal{M}$ denotes an Hidden Markov Model, $\lambda = (A, B, \pi)$ denotes the model parameters.

2 The evaluation problem

Let us formally define the problem: let $\mathcal{M}$ be a given HMM and a $X = x_1 \dots x_T$ an observations sequence, it is requested to calculate the probability p that the automaton $\mathcal{M}$ generates X.

A *brute-force* approach requires to evaluate the probability to obtain X considering all the possible sequences of T length and then sum their probabilities. However an HMM with N states requires to consider N^T possible sequences which is non acceptable for the computational complexity even for trivial models.

An efficient solution for the evaluation problem is given in [1] and it's called *forward and backward procedure*. Let $\alpha_t(i)$ be the observation probability of the sequence $O_1 \cdots O_t$ and let $q_t = S_i$ be the final state of the model λ:

$$\alpha_T(i) = Pr(O_1 O_2 \dots O_T, q_t = S_i|\lambda).$$

The probability of observing just O_1 and the system being at his first step in the state S_i is:

$$\alpha_1(i) = Pr(O_1, q_t = S_i|\lambda) = Pr(O_1|q_t = S_i, \lambda)Pr(q_t = S_i|\lambda) = \pi_i b_i(O_1).$$

Imagine we want to calculate $\alpha_{t+1}(j)$; we can reach the state q_j at time $t+1$ from every other state at time t with probability a_{ij}, thus we have the following recursive relation:

$$\alpha_{t+1}(j) = \Big[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\Big] b_j O_{t+1} \quad 1 \leq t \leq T-1 \wedge 1 \leq j \leq N.$$

In this way we can calculate $P(O|\lambda)$, i.e. the sum of the probabilities to recognise the sequence for every final state:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

The algorithm computational complexity is N^2T.

To solve the first problem we can also use the *backward procedure*. The recursive relation is similar to the one given and the computational complexity is

[5] Note the definition is slightly different from the previous but equivalent

obviously the same. If $\alpha_t(i)$ was the observation probability of the first t symbols and terminating in the state i, $\beta_t(i)$ is the probability of observing the last t symbols of the sequence given the state i:

$$\beta_t(i) = P(O_{t+1} \cdots O_T | q_t = S_i, \lambda).$$

So the recursive relation, is:

$$\beta_t(i) = \begin{cases} 1 & t = T \ \wedge \ 1 \leq i \leq N \\ \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) & 1 \leq t \leq T-1 \ \wedge \ 1 \leq i \leq N \end{cases}$$

3 Finding the optimal state sequence

Formally, the problem requires to find the optimal state sequence Q which maximizes $P(Q|O, \lambda)$ given the observation sequence O and the model λ. Note that:

$$P(Q|O, \lambda) = \frac{P(Q, O|\lambda)}{P(O|\lambda)}.$$

It follows that the problem is equivalent to the maximization of $P(Q, O|\lambda)$. Let $\gamma_t(i)$ be the probability to find the model λ in the state i at time t while the sequence O is produced. Then:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)},$$

thus the solution for the given problem is, locally, for every t:

$$q_t = \arg\max_{1 \leq i \leq N} [\gamma_t(i)].$$

Unlukily this approach works just locally, i.e. it allows one to find the state q_t which maximizes the probability of observing O_t, but we are not sure that obtaining q_t for $1 \leq t \leq T$ we get a sequence which is acceptable for the model λ (for example some a_{ij} could be zero-components).

The Viterbi algorithm is a solution for the problem of maximization which consider only acceptable sequences, and it is introduced in [5]. This algorithm is very similar to the *forward procedure* with the main difference of a recursive maximization of the paths. Let $\delta_t(i)$ be the probability of the optimal state sequence which produced $O_1, \cdots, O_t$ terminating with $q_t = i$, then we can state the following recursive relation:

$$\delta_{t+1} = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1}),$$

where $\delta_1(i) = \pi_i b_i(O_1)$. In this way we can calculate both the probability of the optimal state sequence and the effective state sequence by keeping track in a vector of the choices done for maximizations.

4 HMM training

The third problem concerns the training of a Hidden Markov Model. We have to estimate the parameters of the model $\lambda = (A, B, \pi)$ which maximise $P(O|\lambda)$ given a finite sequence of observation symbols O (i.e. the training set).

We can achieve the result by descent gradient methods obtaining a local minimum (maximum). The computation of a gradient method is often hard, so the descent is usually approximated by specific algorithms such as the *random direction descent* or the *line search* described for example in [9].

The most common algorithm, which just gives local minimum (maximum), is based on *Expectation-Maximization* or simply *EM*, and it is called *Baum-Welch* introduced in [4].

Let $\xi_t(i, j)$ be the probability to find the model in the state S_j at time t and the state S_j al time $t + 1$, given the model and the observation sequence. Formally:

$$\xi_t(i, j) = Pr(q_t = S_i, q_{t+1} = S_j|O, \lambda).$$

We can express this probability as:

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)},$$

where $P(O|\lambda) = \sum_i \sum_j \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$. For the definition of $\gamma_t(i)$ we can state the relation:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

We can iteratively estimate the parameters observing that $\sum_{t=1}^{T-1} \gamma_t(i)$ can be seen as the average number of visits at the state i, while $\sum_{t=1}^{T-1} \xi_t(i, j)$ is the number of transitions from the state S_i to the state S_j. Then we obtain the parameters for the model λ' as:

$$\begin{aligned} \pi' &= \gamma_1(t) \\ a'_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ b'_j(k) &= \frac{\sum_{t=1 \wedge O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned}$$

It is possible to prove that the iterative application of this method converges to a local maximum.

When we have a very complex model, the training algorithm performace can be improved by using the *Viterbi Learning*: both for the gradient methods and EM ones, the main idea is to consider just one path, or a group of paths, instead of all possible paths.

However the *Viterbi Learning* is often not very useful because the improvement of the performances (in practice it is possible to double the calculation

speed) leads to a much lower result quality than the one obtained by the analysis of all paths.

5 Overview on HMM applications

HMM are often used for *speech recognition* purposes as shown in [10] and [2], and for bioinformatic purposes. There are some attempts of using them for images classification purposes but the computational complexity of the algorithms and the obtained results lead us to think that there is still much research to do in this field (for example with modifications to the algorithms to fit new needs). In [8] the authors show that the bi-dimensional classification with HMM does not suffer the problems of local minimum or maximum[6] as much as algorithms based on CART and VQ.

As regards to bioinformatic, HMM are used for data mining purposes in two ways:

- Database mining and sequence classification
- Pattern discovery

5.1 Introduction

HMM are often used for data mining in bioinformatic. The following sections review the state of the art in this field with a special attention to the recognition and comparison of proteic sequence. Each protein is fully described by an amino acid sequence. The amino acid set can be considered a finite symbols alphabet; however we are not sure that the sequence obtained from biology is without errors, and moreover, if we have two proteins with the same function it can happen that an amino acid takes the place of another. This does not allow an exact algorithm to solve the problems concerned to proteins comparison or recognition and it leads to the use of probabilistic algorithms. There are two probabilistic approaches: probabilistic grammars based algorithm and HMM based algorithm (there are also a few specific algorithms for bioinformatic problems which are able to catch the affinity between amino-acid as input).

5.2 Database Mining and classification

Suppose we have a trained HMM with sequence of a protein or a protein set with same function. Then we can use it to look for *similar* amino acid sequences in a sequences database. This can be done thanks to the assumption said in the introduction, i.e similar function proteins have similar amino acid sequence. In this case the required steps are the following:

[6] This means a classification with a low rate of noise. On the other hand HMM based algorithms can hit less details

– Training the model with well-known amino acid sequence of proteins belonging to the same family (i.e. with similar function). EM algorithm can be used for this purpose.
– Searching in the database with amino acid sequences (or part of them) corresponding to unknown proteins, the ones similar to the training set. Viterbi algorithm can be used for this purpose.

Note that the use of the Viterbi algorithm give an affinity probability which tends lo be lower as the analysed sequence becomes longer: this is not a problem if all sequences have the same length but often this is not true. In these cases a formula which weights the Viterbi score on the length of the sequence is used. The formula depends on the analysed data.

As regards the classification issues, it is possible to train a model for each well-known proteins family and then to associate to each sequence a score which basically is the probability that the sequence is part of a protein family. In this case we should have a training set which must be wide enough to carry this classification on with good results. If this is not the case, we can use *unsupervised* clustering algorithms and then train the models on the extracted clusters. Finally we consider an hybrid method which is successfully used in bioinformatic. Suppose we have N classes of objects in a database, where N is unknown, and suppose we have a training set which represents only M classes of objects. Consider without loss of generality $M < N$. We can train the HMMs on the known classes, thus we are able of course to classify the objects in the database on the well-known classes but also group the remaining sequences according to the normalized score respect to each model; the main idea is that if two amino acid sequences are far in a similar way from the well-known classes there's a high probability that they belong the same new class. The algorithms based on this method work better when N is not much greater than M.

The results obtained in bioinformatic are very good (see for example [9]) especially because it seems that HMM works well with the specific structure of analysed data, i.e. the amino acid sequences building the proteins.

In the report [11] the authors compare HMM based algorithms performances with the well-known *Learning Vector Quantization* and *Template Matching* performances: they work on a generic bi-dimensional pattern recognition on images. In this test, the models are trained on the same training set which was treated with K-Mean, and then they study the error rate in function of the introduced noise. The result is that the performances are almost the same for the three algorithms when the noise level is low, but for increasing rate of it, HMM algorithms are less efficient than the other two.

5.3 Two examples taken from Biology

In this section we try to answer this question: why are HMM very appreciated in bioninformatic when other algorithms with same functions seem to have better performance?

Let us consider two examples. The first one is very simple, taken from the work of Churchill [3]: DNA consists of a sequence of bases which can be labelled with the elements of the set A, C, G, T. In many sequencing analysis it can be noted that there are some DNA segments where the bases A-T (state 1) are strongly present, and other ones where the bases G-C are strongly present. The interesting problem is to understand whether we are in the state 1 or 2 among a DNA sequence. Churchill developed, on the base of experimental data and other biological information the following model (see 1):

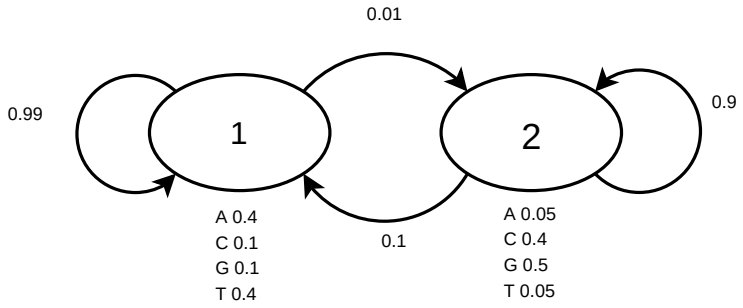

Fig. 1. HMM for the DNA classification made by Churchill. Initial and final states are omitted as in the original work.

In this case the Hidden Markov Model is useful because it allows the researcher to combine the experimental results and other theoretical notions in an easy understandable model which has a strong foundations and can be directly manipulated by well-known algorithms.

The second example shows that HMM are useful when the researcher needs to have a strong control on the intersection penalties or the deletion penalties on an amino acid sequence analysis. Intuitively, by the use of HMM, we can treat each different variation on the pattern in a specific way. As already said, a protein is coded by amino acids: these are an alphabet of twenty symbols. For biological reasons, or for sampling errors, two proteins with the same function can differs in their sequence for:

- the substitution of an amino acid with a similar one (biologists have tables which give the affinity score of amino acids)
- insertion of an amino acid in the sequence
- deletion of an amino acid in the sequence

Of course, two proteins which belongs to the same family are expected to have less differences than two proteins belonging to different families. Figure 2 shows the model introduced in [7] to recognise amino acid sequences. For each matching the authors define a deletion state and an insertion state. For each transition

leading to these states they set a low probability; Finally for the matching states they define the amino acid emission probabilities for that proteins family. In this way for each new state of the sequence we have to add 49 parameters (20 for the emitting probabilities for the state M, 20 for the state D, and 9 for the transition probabilities). This model is very flexible: for example we can represent the fact that insertions at the beginning of the sequence are not important as the ones in the middle (because they could represent an alignment problem) assuming not very low probabilities for the transition leading to $I0$.

In [7] the authors show the experimental results given by the application of the model and the results of the training.

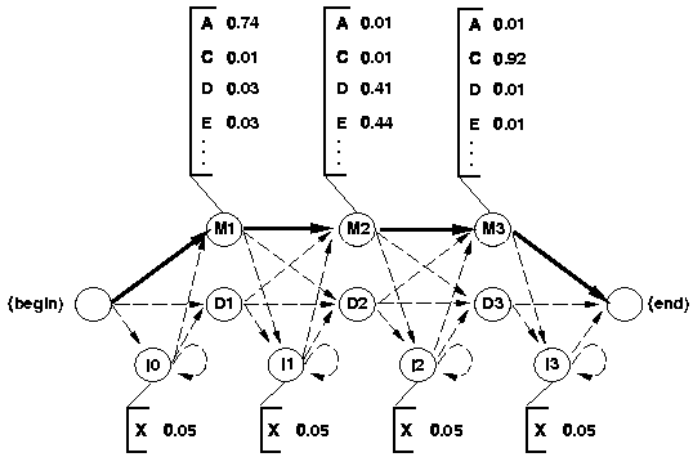

Fig. 2. HMM introduced by Krogh et al. to recognise and classify proteins.

5.4 Pattern discovery

The previous section topics did not take in exam an important matter. How do we set up the initial structure of a HMM? We know that the training can modify the parameters of the model, i.e. the transition probabilities, the symbol emitting probabilities and the initial distribution. We have the need to understand how to build the initial model structure (see for example the number of states) and its initial parameterization[7].

[7] Remember that both descent gradient based algorithm and EM based algorithm can find local minimum or maximum thus they are influenced by the initial state

To understand the relation between this note and the pattern discovery problem, we have to go back to the bioinformatic field where these techniques are successfully used. In fact biologists know some protein structures; this means that they do not know the full amino acid sequence but they know the way they alternate. For example there usually is an initial segment of amino acids, then some sequences which are repeated more times and a final segment. Among this regularity, the amino acid sequences can be quite different and they determine the protein function. Similar proteins have similar sequences. The protein structure is very important for the structure of the HMM; so the result of the application of the training algorithm starting from a well-known model, gives us information on the similarity score between the protein structures; however this job must be supervised.

5.5 Conclusion

The main benefits which data mining can obtain by the use of HMM are related to their solid stochastic-theoretical foundation which stays behind the known algorithms and their efficiency. Moreover these models are useful in the following contexts, and they are essential for many bioinformatic applications:

- they can perform sequence matching for sets where elements differs for insertions, deletions and they can evaluate the associated penalty;
- the model training can be unsupervised;
- they can work with variable length sequences;
- they can be used for alignment purposes, data mining and classification, structural analysis and pattern discovery;
- they can be used for recognition purposes in hierarchical processes.

HMM applications have two main limits. The first one concerns the growth of the number of parameters of the model when the number of the states or the input alphabets grows. This leads to bad algorithm performances due to time issues and bad results depending on the growth of local maximum and minimum.

The second limit is that the emission function and the transition function are independent and both depends only on the hidden state. A typical example of a correlation which can't be caught by a HMM is the following: suppose that the emission of the symbol X in the state S_i is often followed by the emission of the symbol Y in the state S_j and that X' in the state S_i is often followed by Y' in the state S_j: it is clear that this situation cannot be represented by a first order Markov chain and consequently disinformation is lost in the HMM analysis.

References

1. Baum, L. E., and Egon, J. A. An inequality for rational functions with applications to somestatistical estimation problems. *Bull. Amer. Metereol. Soc. 73* (1968), 211–227.
2. Bourlard, H., and Bengio, S. Hidden markov models and other finite state automata for sequence processing. *Idiap Research Report* (2001).

3. Churchill, G. A. Stochastic models for etherogeneus dna sequences. *Bull Math Biol 51* (1989), 79–94.
4. Dempster, A., Laird, N., and Rubin, D. Maximun likelihood from incomplete data via em algorithm. *J. Roy. Stat. Soc. 39*, 1 (1977), 1–38.
5. Forney, G. D. The viterbi algorithm. *Proc. IEEE 61* (1973), 268–278.
6. Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*, second ed. Addison-Wesley, 2000.
7. Krogh, A., Brown, B., Mian, I. S., Sjolander, K., and Haussler, D. Hidden markov models in computational biology: applications to protein modeling. *J Mol Biol 235* (1994), 1501–1531.
8. Li, J. S. U., Najmi, A., and Gray, R. M. Image classification by a two-dimensional hidden markov model. *IEEE Transactions on Signal Processing 48*, 2 (2000), 517–533.
9. P. Baldi, S. B. *Bioinformatics: the machine learning approach*, second ed. MIT Press, 2001.
10. Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989).
11. Shahram, M., and Milanfar, P. Hmm-based pattern detection. *Project Report for EE 262: Image Processing and Reconstruction* (2002).