

PAC-Learning of Markov Models with Hidden State

Doina Precup

Joint work with Ricard Gavalda (Spain), Joelle Pineau, Philipp Keller

Reasoning and Learning Laboratory

School of Computer Science

McGill University

Published in ECML'06

Motivation

- *Learning probabilistic models for time series / sequence data*
- Standard framework: Hidden Markov Models, trained with Expectation Maximization
 - + General formulation
 - + Many success stories (e.g. speech, bioinformatics)
 - Need to know the set of hidden (latent states)
 - Convergence to a locally optimal solution, with quality often dependent on initial parameters

Alternative frameworks

- Predictive representations:
 - Predictive State Representations (Littman et al., NIPS'02, Singh et al, UAI'04, Rosencrantz et al., 2004)
 - Observable Operator Models (Jaeger, 2000)
 - Diversity-based representations (Rivest & Schapire, 1994)
 - Predictive Linear Gaussian models (Wingate & Singh, 2006)
- History-based models
- Boltzmann machines

Our approach: based on the work on PAC-learning of finite automata

Outline

- Notation
- *Approximating HMMs* with Probabilistic Deterministic Finite Automata (PDFAs)
- *Learning algorithm for PDFAs*
 - Infers both the state representation and the parameters
 - Formal PAC-style performance guarantees
- Small empirical illustrations

Hidden Markov Models (HMMs)

- Finite set of states S
- Finite set of observations Z
- Initial belief distribution b_0
- Transition function $T : S \times S \rightarrow [0, 1]$

$$T(s, s') = P(s_{t+1} = s' | s_t = s)$$

- Emission function $O : S \times Z \rightarrow [0, 1]$

$$O(s, z) = P(z_t = z | s_t = s)$$

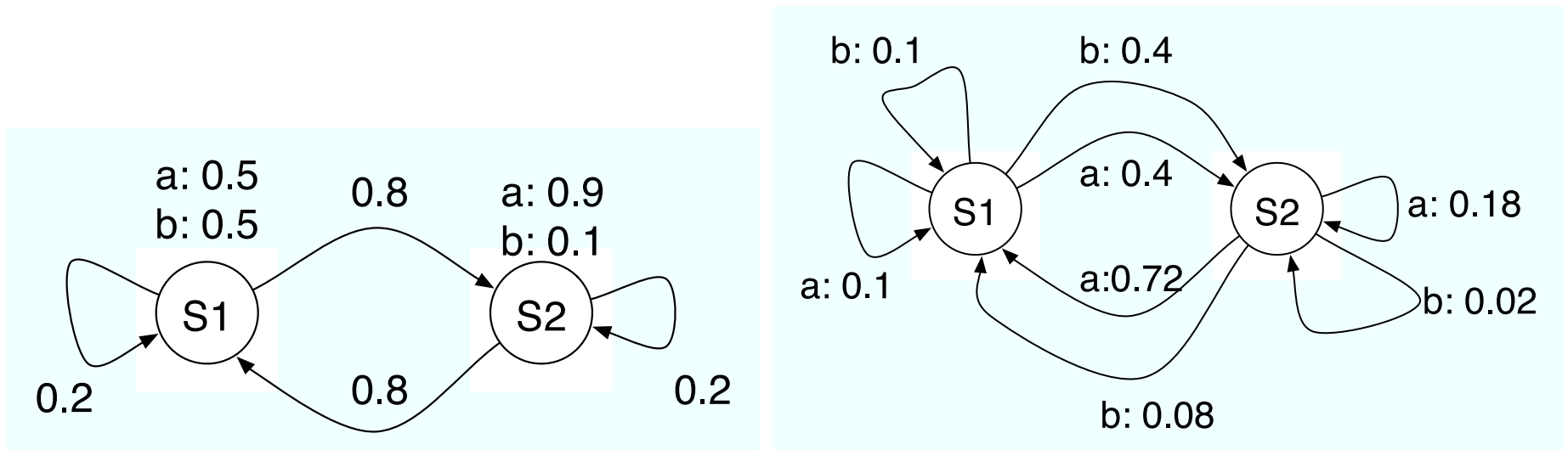
Probabilistic Nondeterministic Finite Automaton (PNFA)

- Finite set of states S
- Finite set of observations Z
- Initial belief distribution b_0
- Transition function $T : S \times Z \times S \rightarrow [0, 1]$

$$T(s, z, s') = P(s_{t+1} = s' | s_t = s, z_t = z)$$

- *Emissions on transitions instead of states*

Transforming an HMM into an equivalent PNFA

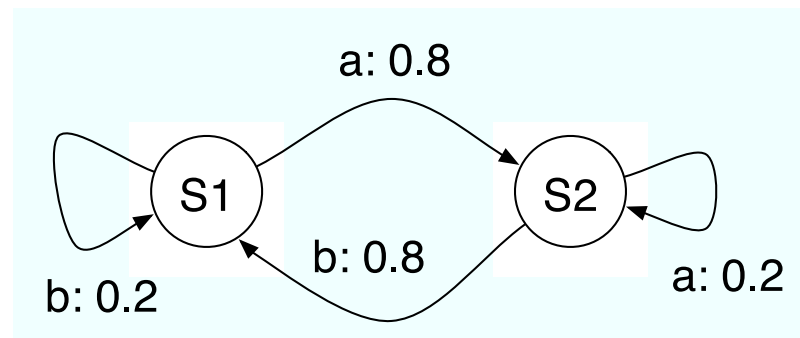


- Abe & Warmuth, 1992: For any HMM, there is an *equivalent PNFA* with the same number of states
- Equivalent = produces same probability distribution on observation sequences
- Dupont et al, 2005: A hairier construction to transform a PNFA with n states into an HMM with n^2 states

Probabilistic Deterministic Finite Automaton (PDFA)

- Finite set of states S
- Finite set of observations Z
- Initial belief distribution b_0
- Transition function $T : S \times Z \rightarrow S, T(s, z) = s'$
Unique next state once the observation is selected
- Emission function $O : S \times Z \rightarrow [0, 1]$

$$O(s, z) = P(z_t = z | s_t = s)$$



Transformations between PDFAs and HMMs

- Any PDFA with n states can be represented by an HMM with n^2 states
 - A state in the HMM corresponds to a pair of states in the PDFA
 - Observations and transitions are constructed in the obvious way
- *Not every PNFA can be represented by an equivalent PDFA*
- But every PNFA can be *approximated by a PDFA*

Measuring dissimilarity of probability distributions

For two distributions P_1, P_2 :

$$L_\infty(P_1, P_2) = \max_x |P_1(x) - P_2(x)|$$

$$D_{KL}(P_1, P_2) = \sum_x P_1(x) \log \frac{P_1(x)}{P_2(x)}$$

Approximation Result

For every PNFA M and every $\epsilon > 0$ there is a finite-size PDFFA that approximates M with precision ϵ in L_∞ distance.

Main idea: there are finitely many strings with probability greater than ϵ , so construct a tree-shaped PDFFA with one leaf for every such string.

This is a theoretical construction, not what you would do in practice

Action plan

- Instead of learning an HMM, we want to learn a PDFA which approximates within a desired precision ϵ
- We want PAC-style guarantees for the algorithm

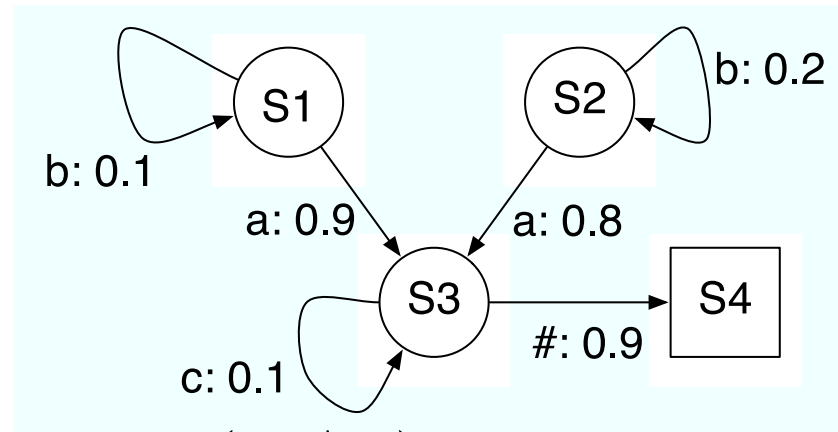
Learnability

- Data is in the form of observation sequences drawn iid from the true distribution
- A model is called ϵ -good if the distance between its trajectory distribution and the distribution produced by the true system is within ϵ (where distance could be L_∞ or D_{KL})
- If we would have a learning algorithm such that, for any PDFA M , we obtain an approximation M' that is ϵ -good with probability at least $1 - \delta$, in time $\text{poly}(n, 1/\epsilon, 1/\delta)$, we would say PDFAs are efficiently PAC-learnable
- Similar definition for sample complexity
- Kearns et al., 1994: the class of *all PDFAs* is *not efficiently PAC-learnable*

What kind of PDFAs can be learned?

- Ron et al, 1995: Some PDFAs are efficiently learnable
 - PDFAs must be acyclic
 - States must be *distinguishable*
- Two states s and s' are called *μ -distinguishable* if $L_\infty(D(s), D(s')) \geq \mu$, where $D(s)$ is the probability distribution over strings generated from s
- A PDFA is *μ -distinguishable* if every two states are *μ -distinguishable*
- Clark & Thollard, 2004: Algorithm that works for cyclic automata in sample size $\text{poly}(n, 1/\epsilon, \log 1/\delta, 1/\mu, L)$, where L is a bound on the expected length of the generated strings

Distinguishability example



$$P(a\#|s1) = 0.81$$

$$P(a\#|s2) = 0.72$$

- All other strings have closer probabilities.
- So S1 and S2 are 0.08-distinguishable but not 0.1-distinguishable
- Intuitively, if we want to distinguish states that are closer, we will need more samples

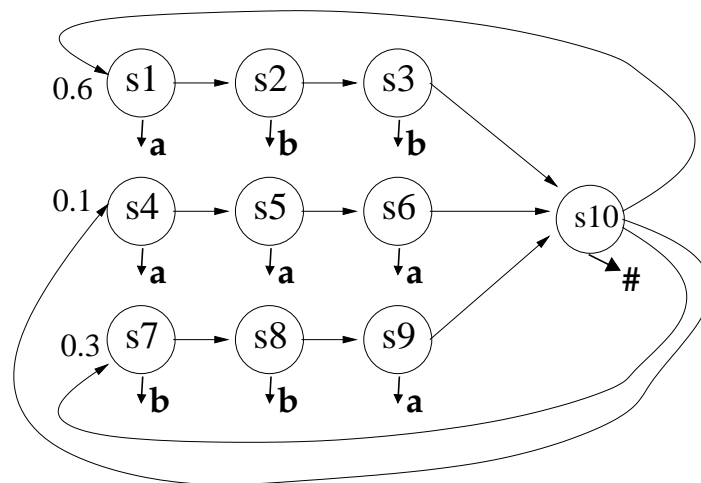
The Clark-Thollard algorithm

- Uses D_{KL} to measure model performance
- Need to specify all parameters upfront - *very hard*
- Based on parameters, compute the number of needed trajectories m and build a PDFA
- For $n = L = 3$, $\epsilon = \delta = \mu = 0.1$, we get $m = 10^{20}$!
- This is due to very conservative analysis

Our approach

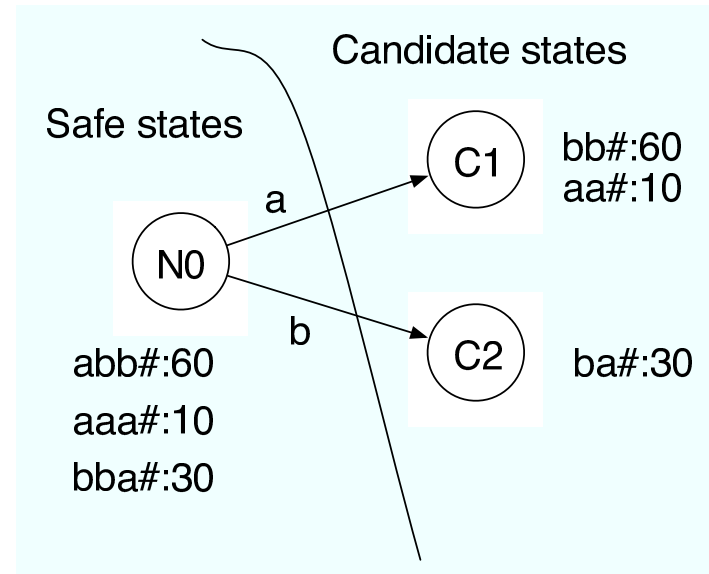
- Give number of trajectories m instead of L, ϵ
- Use L_∞ distance
- The time to get the state structure is separated, smaller for “easier” graphs

Algorithm: Simple example (1)



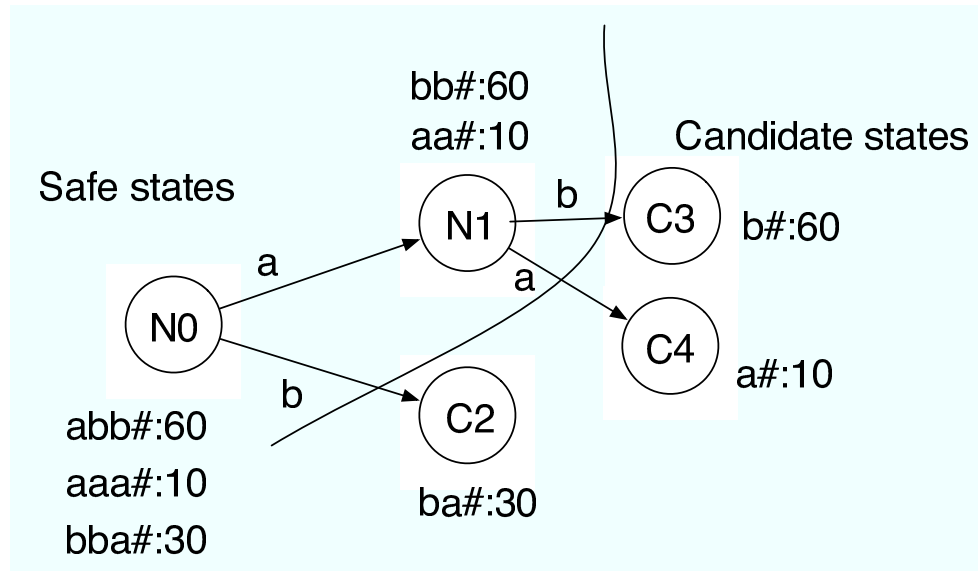
- HMM generates $abb\#$, $aaa\#$ and $bba\#$, with probabilities 0.6, 0.1, 0.3
- Suppose we have m trajectories
- We maintain two categories of states:
 - **Safe states**: states that are quite certain based on the data
 - **Candidate states**: states that are possible, but less certain

Algorithm: Simple example (2)



- One initial safe state
- Two candidate states (one for each symbol present in the data)
- Each state has a set of data, containing the trajectories that start at the state

Algorithm: Simple example (3)



- If a candidate state is *large enough*, decide whether to:
 - *promote* it to be a new safe state
 - *merge* it with an existing safe state
- Continue until no state is large enough

Algorithm outline

1. Define initial safe state, labeled with empty string
2. Define one candidate state for every observation
3. While there are still samples left
 - Run sample through the current graph
 - If it ends at candidate state (s, z) , add the remaining trajectory to the set of samples at this state, $D_{(s,z)}$
 - If $D_{(s,z)}$ is large enough, either promote (s, z) or merge it with an existing safe state
4. Build PDFAs from current graph

Merging and promoting states

Largeness condition: $D_{(s,z)}$ should have at least

$$\frac{c}{\mu^2} \log \frac{n|Z|}{\delta}$$

trajectories

Assuming that the original PDFA is μ -distinguishable

- If the distributions observed at (s, z) and some safe state s' have distance $\leq \mu/2$, identify (s, z) and s'
- Else, promote (s, z) to a new safe state, create new candidate states from it
- Re-distribute the trajectories from $D_{(s,z)}$ to the nodes in the graph

Building PDFA from the graph

- Identify each remaining candidate state with the closest safe state (in terms of the L_∞ distance)
- Compute emission probabilities in the obvious way, using counts and smoothing

Implementation issues

- Algorithm can be implemented both in batch and in on-line mode
- In on-line mode, states can keep getting large as more data is acquired
- Can limit the size of the automaton that will be learned, if desired

Learning the topology

Suppose a state s is reachable by a path of length l whose edges all have probability $\geq p$. Then a corresponding safe state will be created in the graph in time

$$O\left(\frac{l}{\mu^2 p} \log \frac{n|Z|}{\delta}\right)$$

- Time depends on l and p , which are unknown but characterize how easy it is to reach s
- No dependency on ϵ
- Dependency on L only indirectly through p

Learning the parameters

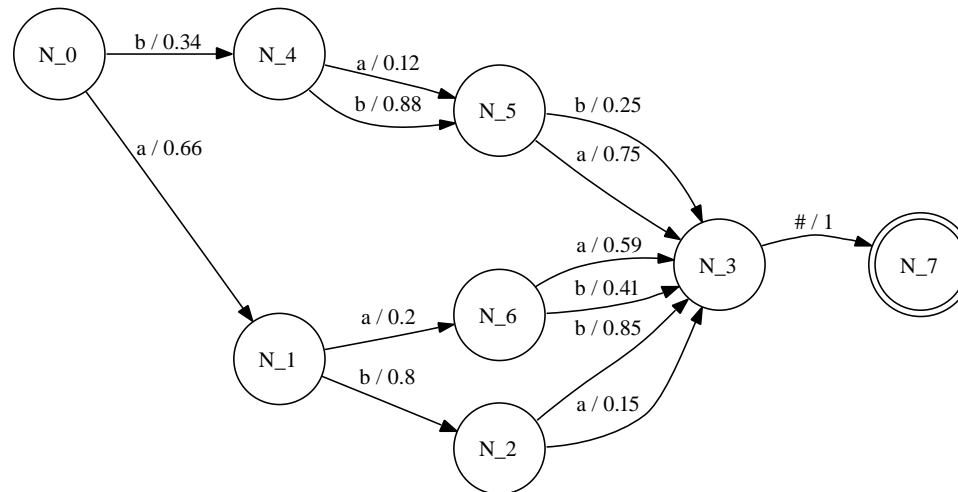
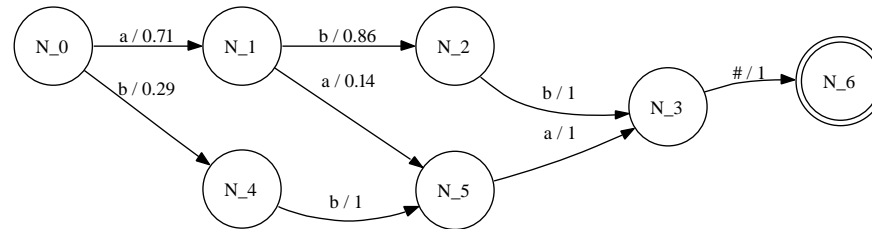
Suppose the graph built is isomorphic to the minimal representation of the target automaton. Then, with:

$$\text{poly} \left(n, \frac{1}{\epsilon}, \log \frac{1}{\delta}, \frac{1}{\mu}, L \right)$$

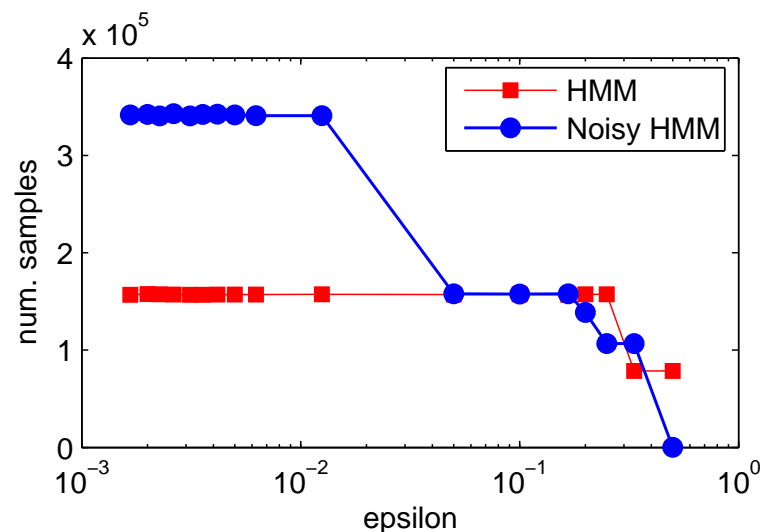
additional samples, the PDFA will satisfy the PAC-learning criterion

Proof is basically same as in Clark & Thollard.

Results: No noise vs Observation noise



Predicted number of samples

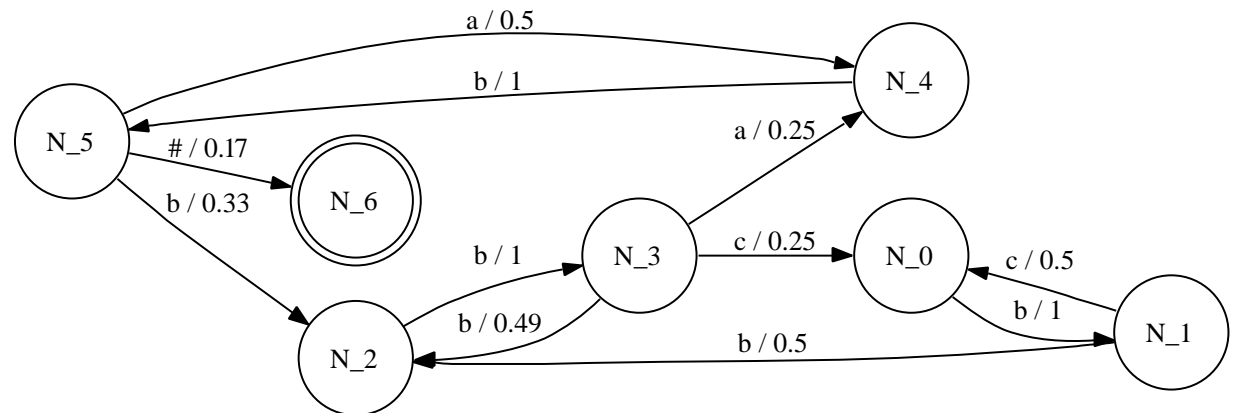


- Noisy case requires more samples
- Number predicted by PAC result much lower than for Clark & Thollard, but still overestimate
- An order of magnitude fewer samples needed in the experiments

Illustration: Cheese maze

- Transitions uniformly random to neighboring states
- Observations: a=1 wall, b = 2 walls, c = 3 walls
- S10 is a terminal state
- Start uniformly randomly at S5 or S7

S0	S1	S2	S3	S4
S5		S6		S7
S8		S10		S9



- Each state of the learned PDFA has a natural interpretation
- E.g. N0 corresponds to the initial belief, N1 to uniform belief in S0 S4, S8 S9

Relationship to other work

- Holmes and Isbell, ICML'06, presented a special case of the analysis and algorithm for deterministic systems
- The automaton constructed approximates the original HMM in terms of trajectory distribution
- Similar philosophy to PSR: we do not care to recover structure, just to be able to predict well trajectory probabilities
- The set of trajectories at every state can be viewed as an observed set of PSR tests
- Unlike PSR learning algorithms, we need orders of magnitude fewer samples, and can provide better guarantees
- This is due to the fact that we *do not compute the set of core tests* (no need to get linearly independent columns)

Conclusions and future work

- A PAC-learning algorithm for learning a PDFA that approximates an HMM
- Learns both structure and transition probabilities
- Much better sample complexity bounds than in existing work

Future work:

- Extend to distances other than L_∞
- Can we avoid specifying μ ?
- Tighter sample complexity analysis
- More experiments!

References

Clark and Thollard, PAC-Learnability of PDFAs. JMLR'04.

Dupont, Denis & Esposito, Links between probabilistic automata and HMMs. Pattern Recognition'05.

Holmes & Isbell. Looping Suffix-Tree based inference of partially observable hidden state. ICML'06

Jaeger, Zhao & Kolling. Efficient estimation of OOMs. NIPS'05 Kearns et al, On the learnability of discrete distributions. ACM Symposium on Theory of Computing, 1995

Ron, Singer & Tishby. On the learnability and usage of acyclic probabilistic finite automata

Rosencrantz, Gordon & Thrun, Learning low dimensional predictive representations, ICML'04

Littman, Sutton & Singh. Predictive representations of State. NIPS'01.