Bayesian Learning

Computer Science 760

Patricia J Riddle

Introduction

Bayesian learning algorithms **calculate explicit probabilities** for hypotheses

Naïve Bayes classifier is among the most effective in classifying text documents

Bayesian methods can also be used to analyze other algorithms

Training example incrementally **increases or decreases** the estimated **probability** that a hypothesis is correct

Prior knowledge can be combined with observed data to determine the final probability of a hypothesis

Prior Knowledge

Prior knowledge is:

1. Prior probability for each candidate hypothesis and

2. A probability distribution over observed data

Bayesian Methods in Practice

Bayesian methods accommodate hypotheses that make **probabilistic predictions** "this pneumonia patient has a 98% chance of complete recovery"

New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities

Even when computationally intractable, they can provide a **standard of optimal decision making** against which other practical measures can be measured

Practical Difficulties

1. Require **initial knowledge of many probabilities** - estimated based on background knowledge, previously available data, assumptions about the form of the underlying distributions

2. **Significant computational cost** to determine the **Bayes optimal hypothesis** in the general case - linear in the number of candidate hypotheses - in certain specialized situations the cost can be significantly reduced

Bayes Theorem Intuition

Learning - we want the best hypothesis from some space H, given the observed training data D. **Best** can be defined as **most probable** given the data D plus any initial knowledge about prior probabilities of the various hypotheses in H.

This is a **direct method**!!! (No Search)

Bayes Terminology

P(h) – the **prior probability** that hypothesis h holds before we observe the training data - prior probability - if we have no prior knowledge we assign the same initial probability to them all (it is trickier than this!!)

P(D) - **prior probability** training data D will be observed given no knowledge about which hypothesis holds

P(D|h) - the probability of observing data D given that hypothesis h holds

P(h|D) - the probability that h holds given the training data D - ***posterior probability***

Bayes Theorem

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

Probability increases with P(h) and P(Dlh) and decreases with P(D) - this last is not true with a lot of other scoring functions!

MAP & ML Hypothesis

So we want a **maximum a posteriori** hypothesis (MAP) - P(D) same for every hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(D \mid h) P(h)$$

If we assume every hypothesis is equally likely a priori then we want the **maximum likelihood** hypothesis

$$h_{ML} \equiv \arg\max_{h \in H} P(D \mid h)$$

Bayes theorem is more general than Machine Learning!!

A general Example

Two hypothesis: the patient has cancer, $\oplus$, the patient doesn't have cancer, $\ominus$

Prior knowledge: over the entire population of people .008 have cancer

The lab test returns a correct positive result in 98% of the cases in which cancer is actually present and a correct negative in 97% of the cases in which cancer is actually not present

$P(\text{cancer}) = .008, P(\neg\text{cancer}) = .992$

$P(\oplus|\text{cancer}) = .98, P(\ominus|\text{cancer}) = .02$

$P(\oplus|\neg\text{cancer}) = .03, P(\ominus|\neg\text{cancer}) = .97$

So given a new patient with a positive lab test, should we diagnose the patient as having cancer or not??

Which is the MAP hypothesis?

Example Answer

Has cancer - P(⊕|cancer)P(cancer) = (.98).008 = .0078

Doesn't have cancer - P(⊕|¬cancer)P(¬cancer)=(.03).992=.0298

h_{MAP}=¬cancer

Exact posterior probabilities –

$$P(cancer \mid \oplus) = \frac{P(\oplus \mid cancer)P(cancer)}{P(\oplus)} = \frac{.0078}{P(\oplus)}$$

Posterior as a real probability

$$\frac{.0078}{P(\oplus \mid cancer) + P(\oplus \mid \neg cancer)} = \frac{.0078}{.0078 + .0298} = .21$$

Minimum Description Length

Let us look at h_{MAP} in the light of basic concepts of information theory

$$h_{MAP} \equiv \text{argmax}_{h \in H} P(D|h)\, P(h)$$

$$= \text{argmin}_{h \in H} - \log_2 P(D|h) - \log_2 P(h)$$

This can be interpreted as a statement that short hypotheses are preferred.

Information Theory

- Consider the problem of designing a code to transmit messages drawn at random, where the probability of encountering message i is p_i.
- We want the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random.
- To minimize the expected code length we should assign shorter codes to more probable messages.

Optimal Code

- Shannon and Weaver (1949) showed the optimal code assigns $-\log_2 p_i$ bits to encode message i. Where p_i is the probability of i appearing.
- $L_C(i)$ is the description length of message i with respect to code C.
- L_{CH} is the size of the description of the hypothesis h using the optimal representation for encoding the hypothesis space H.

$$L_{C_H}(h) = -\log_2 P(h)$$

- $L_{CD|h}$ is the size of the description of the training data D given the hypothesis h using the optimal representation for encoding the data D assuming that both the sender and receiver know the hypothesis h.

$$L_{C_{D|h}}(D \mid h) = -\log_2 P(D \mid h)$$

Applying MDL

- To apply this principle we must choose specific representations C1 and C2 appropriate for the given learning task!
- **Minimum Description Length Principle**:

$$h_{MDL} \equiv \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D \mid h)$$

- ***If*** C1 and C2 are chosen to be optimal encodings for their respective tasks, then $h_{MDL}=h_{MAP}$

MDL Example

- Apply MDL principle to the problem of learning decision trees.
- C_1 is an encoding of trees where the description length grows with the number of nodes in the tree and the number of edges.
- C_2 transmits misclassified examples by identifying
  - which example is misclassified ($\log_2 m$ bits, where m is the number of training instances) and
  - its correct classification ($\log_2 k$ bits, where k is the number of classes).

MDL Intuition

MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis

So the MDL principle produces a MAP hypothesis if the encodings C_1 and C_2 are optimal. But to show that we would need all the prior probabilities P(h) as well as P(D| h).

No reason to believe the MDL hypothesis relative to arbitrary encodings should be preferred!!!!

What I hate about MDL

"But you did't find the optimal encodings C1 and C2."

"Well it doesn't matter if you see enough data it doesn't matter which one you use."

So why are we using a Bayesian approach????

Bayes Optimal Classifier

What is the most probable classification of the new instance given the training data?

Could just apply the MAP hypothesis, but can do better!!!

Bayes Optimal Intuitions

Assume three hypothesis h1, h2, h3 whose posterior probabilities are .4, .3 and .3 respectively.

Thus h1 is the MAP hypothesis.

Suppose we have a new instance x which is classified positive by h1 and negative by h2 and h3.

Taking all hypothesis into account, the probability that x is positive is .4 and the probability it is negative is .6.

The most probable classification (negative) is different than the classification given by the MAP hypothesis!!!

Bayes Optimal Classifier II

We want to combine the predictions of all hypotheses weighted by their posterior probabilities.

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

where v_j is from the set of classifications V.

Bayes Optimal Classification:

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

No other learner using the **same hypothesis space** and **same prior knowledge can outperform** this method **on average**. It maximizes the probability that the new instance is classified correctly.

Gibbs Algorithm

Bayes Optimal is quite costly to apply. It computes the posterior probabilities for every hypothesis in H and combines the predictions of each hypothesis to classify each new instance.

An alternative (less optimal) method:

1. Choose a hypothesis h from H at random, according to the posterior probability distribution over H.
2. Use h to predict the classification of the next instance x.

Under certain conditions the expected misclassification error for Gibbs algorithm is at most twice the expected error of the Bayes optimal classifier.

What is Naïve Bayes?

Results comparable to ANN and decision trees in some domains

Each instance x is described by a conjunction of attribute values and the target value f(x) can take any value from a set V. A set of training instances are provided and a new instance is presented and the learner is asked to predict the target value.

$$V_{MAP} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2 \ldots a_n)$$

$$= \arg\max_{v_j \in V} P(a_1, a_2, \ldots a_n \mid v_j) P(v_j)$$

$P(v_j)$ is estimated by the frequency of each target value in the training data.

Cannot use frequency for $P(a_1, a_2, \ldots a_n \mid v_j)$ unless we have a very, very large set of training data to get a reliable estimate.

Conditional Independence

Naïve Bayes assumes attribute values are conditionally independently given the target value - $P(a_1, a_2, ... a_n \mid v_j) = \prod_i P(a_i \mid v_j)$

Naïve Bayes Classifier:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

where v_{NB} denotes the target values

$P(a_i|v_j)$ can be estimated by frequency

When is Naïve Bayes a MAP?

When conditional independence assumption is satisfied the naïve Bayes classification is a MAP classification

Naïve Bayes entails no search!!

An Example

Target concept PlayTennis

Classify the following instance: <Outlook=sunny, Temperature = cool, Humidity = high, Wind = strong>

$$v_{NB} = \arg\max_{v_j \in \{yes,no\}} P(v_j)P(Outlook = sunny \mid v_j)P(Temperature = cool \mid v_j)$$
$$P(Humidity = high \mid v_j)P(Wind = strong \mid v_j)$$

P(PlayTennis=yes)=9/14=.64

P(PlayTennis=no)=5/14=.36

P(Wind=strong|PlayTennis=yes)=3/9=.33

P(Wind=strong|PlayTennis=no)=3/5=.60

An Example II

P(yes) P(sunny|yes) P(cool|yes) P(high|yes) P(strong|yes) = .0053

P(no) P(sunny|no) P(cool|no) P(high|no) P(strong|no) = .0206

Naïve Bayes returns "Play Tennis = no" with probability

$$\frac{.0206}{.0206 + .0053} = 0.7954 = 79.5\%$$

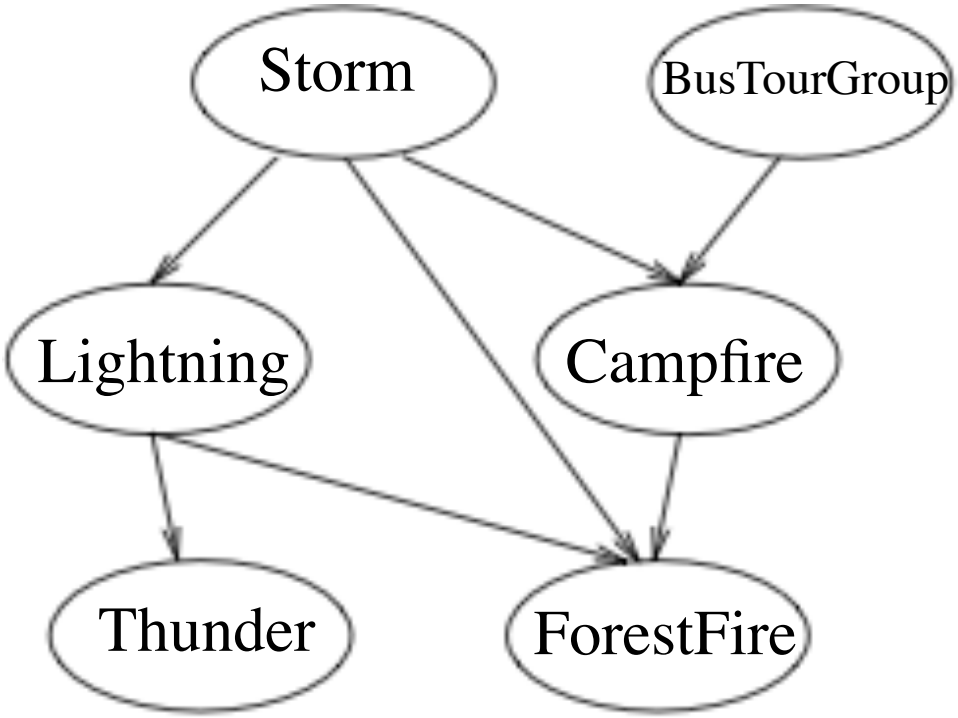
Naïve Bayes used for Document Clustering

- Are the words conditionally independent?
- Works really well anyway

Bayesian Belief Networks

- Naïve Bayes assumes all the attributes are conditionally independent
- Bayesian Belief Networks (BBNs) describe a joint probability distribution over a set of variables by specifying a set of conditional independence assumptions and a set of conditional probabilities
- X is conditionally independent of Y means P(X|Y,Z) = P(X|Z)

A Bayesian Belief Network



	S,B	S,¬B	¬S,B	¬S,¬B
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Campfire

Representation

- Each variable is represented by a node and has two types of information specified.
  1. Arcs representing the assertions that the variable is conditionally independent of its nondescendents given its immediate predecessors (I.e., Parents). X is a descendent of Y if there is a directed path from Y to X.
  2. A conditional probability table describing the probability distribution for that variable given the values of its immediate predecessors. This joint probability is computed by

$$P(y_1,\ldots,y_n) = \prod_{i=1}^{n} P(y_i \mid Parents(y_i))$$

Representation II

- Campfire is conditionally independent of its nondescendents Lightning and Thunder given its parents Storm and BusTourGroup

$$P(Campfire = True \mid Storm = True, BusTourGroup = True) = 0.4$$

- Also notice that ForestFire is conditionally independent of BusTourGroup and Thunder given Campfire and Storm and Lightning.
- Similarly, Thunder is conditionally independent of Storm, BusTourGroup, Campfire, and ForestFire given Lightning.
- BBNs are a convenient way to represent causal knowledge. The fact that Lightning causes Thunder is represented in the BBN by the fact that Thunder is conditionally independent of other variables in the network given the value of Lightning.

Inference

- Can we use the BBN to infer the value of a target variable ForestFire given the observed values of the other variables.
- Infer not a single value but the probability distribution for the target variable which specifies the probability it will take on each possible value given the observed values of the other variables
- Generally, we may wish to infer the probability distribution of a variable (e.g., ForestFire) given observed values for only a subset of the other variables (e.g., Thunder and BusTourGroup are the only observed values available).
- Exact inference of probabilities (and even some approximate methods) for an arbitrary BBN is known to be NP-hard.
- Monte Carlo methods provide approximate solutions by randomly sampling the distributions of the unobserved variables

Learning BBNs

- If the network structure was given in advance and the variables are fully observable, then just use the Naïve Bayes formula modulo only some of the variables are conditionally independent.
- If the network structure is given but only some of the variables are observable, the problem is analogous to learning weights for the hidden units in an ANN.
- Similarly, use a gradient ascent procedure to search through the space of hypotheses that corresponds to all possible entries in the conditional probability tables. The objective function that is maximized is P(D|h).
- By definition this corresponds to searching for the maximum likelihood hypothesis for the table entries.

Gradient Ascent Training of BBN

- Let w_{ijk} denote a single entry in one of the conditional probability tables. Specifically that variable Y_i will take on value y_{ij} given that its parents U_i take on the values u_{ik}.
- If w_{ijk} is the top right entry, then Y_i is the variable Campfire, U_i is the tuple of parents <Storm, BusTourGroup>, y_{ij}=True and u_{ik}=<False,False>.
- The derivative for each w_{ijk} is

$$\frac{\partial \ln P(D \mid h)}{\partial w_{ij}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik} \mid d)}{w_{ijk}}$$

Weight Updates

- So back to our example we must calculate P(Campfire = True, Storm = False, BusTourGroup = False | d) for each training example d in D. If the required probability is unobservable then we can calculate it from other variables using standard BBN inference.
- As weights w_{ijk} are updated they must remain in the interval [0,1] and the sum $\Sigma_j w_{ijk}$ remains 1 for all i,k. So must have a two step process.

1. $$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{w_{ijk}}$$
2. Renormalize the weights w_{ijk}

- Will converge to a locally maximum likelihood hypothesis for the conditional probabilities in the BBn.

Summary

- Bayesian methods provide a basis for probabilistic learning methods that accommodate knowledge about prior distributions of alternative hypothesis and about the probability of observing the data given various hypothesis. They assign a posterior probability to each candidate hypothesis, based on these assumed priors and the observed data,
- Bayesian methods return the most probable hypothesis (e.g., a MAP hypothesis).
- Bayes Optimal classfier combines the predictions of all alternative hypotheses weighted by their posterior probabilities, to calculate the most probable classification of a new instance.

Naïve Bayes Summary

- Naïve Bayes has been found to be useful in many killer apps.
- It is naïve because it has no street sense….no no no…it incorporates the simplifying assumption that attribute values are conditionally independent given the classification of the instance.
- When this is true naïve Bayes produces a MAP hypothesis.
- Even when the assumption is violated Naïve Bayes tends to perform well.
- BBNs provide a more expressive representation for sets of conditional independence assumptions.

Minimum Description Length Summary

- The Minimum Description Length principle recommends choosing the hypothesis that minimizes the description length of the hypothesis plus the description length of the data given the hypothesis.
- Bayes theorem and basic results from information theory can be used to provide a rationale for this principle.

Hidden Markov Models

Comp Sci 369

Dr Patricia Riddle

Automata Theory

An automaton is a mathematical model for a **finite state machine** (FSM).

An FSM is a machine that, given an input of symbols, '**jumps**' through a series of states according to a **transition function** (which can be expressed as a table).

Finite State Machine

A model of computation consisting of a set of ***states***, a ***start state***, an input ***alphabet,*** and a ***transition function*** that maps input symbols and current states to a next state.

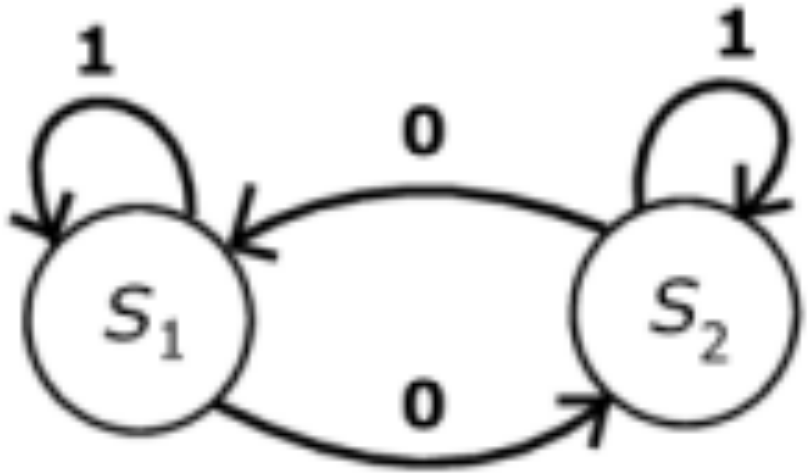
Computation begins in the start state with an input string. It changes to new states depending on the transition function.

Also known as finite state automaton

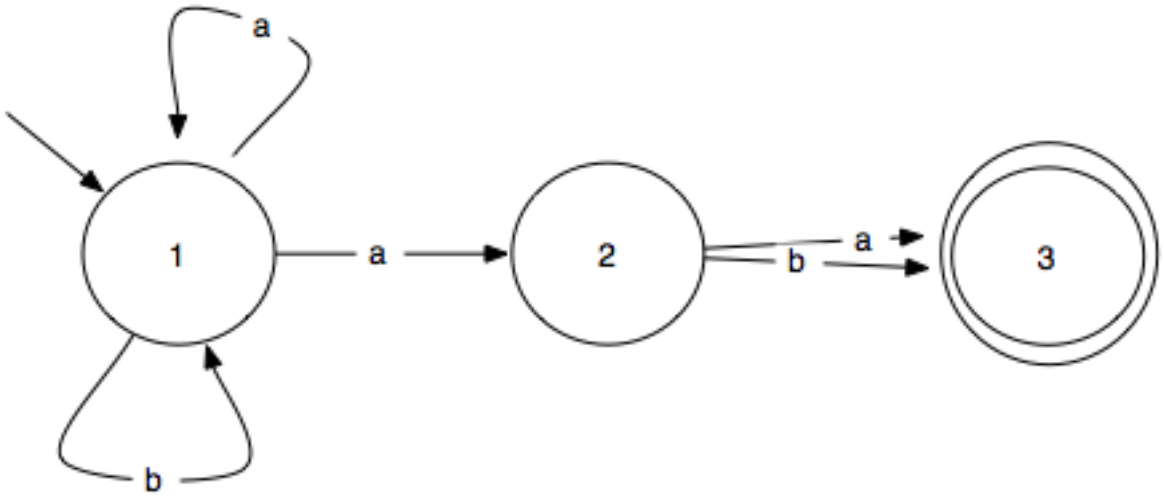
Transition

Current State/ Condition	State A	State B	State C
Condition X	…	…	…
Condition Y	…	State C	…
Condition Z	…	…	…

Deterministic Finite Automata



Nondeterministic Finite Automata



Variations

There are many variants, for instance,

machines having actions (outputs) associated with

transitions (*Mealy machine*) or

states (*Moore machine*),

multiple start states,

transitions conditioned on no input symbol (a null)

more than one transition for a given symbol and state (***nondeterministic finite state machine***),

one or more states designated as *accepting states* (*recognizer*), etc.

Finite State Machines

An **automaton** is represented by the 5-tuple $<Q,\Sigma,\delta,q_0,F>$, where:

Q is a set of **states**.

Σ is a finite set of *symbols*, that we will call the **alphabet** of the language the automaton accepts.

δ is the **transition function**, that is $\delta: Q\times\Sigma\rightarrow Q$ (For non-deterministic automata, the empty string is an allowed input).

q_0 is the **start state**, that is, the state in which the automaton is when no input has been processed yet (Obviously, $q_0 \in Q$).

F is a set of states of Q (i.e. $F \subseteq Q$), called **accept states**.

Markov Chains - CS Definition

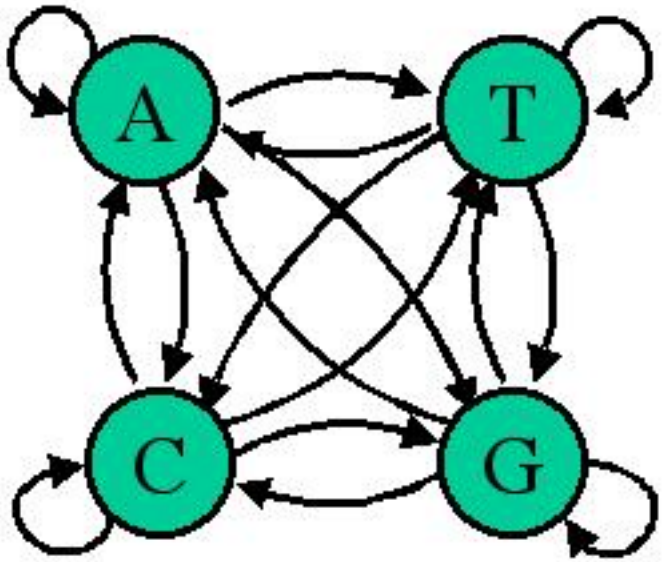
Markov chain - A ***finite state machine*** with probabilities for each transition, that is, a probability that the next state is s_j given that the current state is s_i.

Note: Equivalently, a **weighted, directed graph** *in which the weights correspond to the probability of that transition.*

In other words, the weights are nonnegative and the total weight of outgoing **edges** *is positive.*

If the weights are normalized, the total weight, including self-loops, *is 1.*

Markov chain for DNA sequence



$$P(ATCGCGTA...) = \pi_A a_{AT} a_{TC} a_{CG} a_{GC} a_{CG} a_{GT} a_{TA} \ldots$$

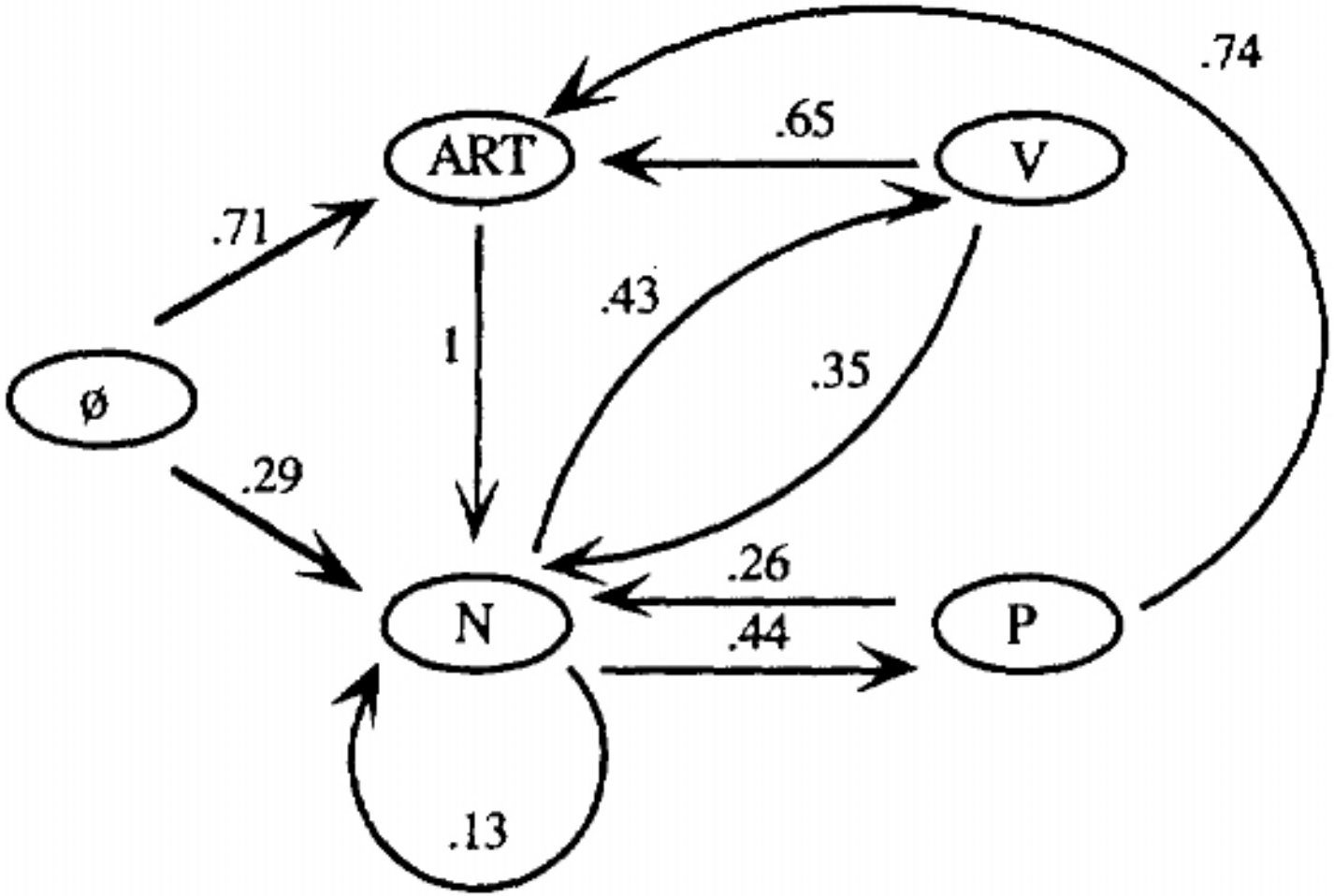


Figure 7.7 A Markov chain capturing the bigram probabilities

Markov Chains - Statistics Definition

In mathematics, a **Markov chain**, named after Andrey Markov, is a **discrete-time stochastic process** with the **Markov property**.

That is a Markov chain is a series of states of a system that has the **Markov property**.

At each time the system may have changed from the state it was in the moment before, or it may have stayed in the same state. The changes of state are called **transitions**.

Markov Property

If a sequence of states has the **Markov property**, it means that every future state is **conditionally independent** of every prior state given the current state.

Chain rule:

$$P(w_1,\ldots,w_n) = P(w_1)\prod_{i=2}^{n} P(w_i \mid w_1,\ldots,w_{i-1})$$

Markov assumption:

$$P(w_1,\ldots,w_n) \approx P(w_1)\prod_{i=2}^{n} P(w_i \mid w_{i-1})$$

Conditionally Independent

In probability theory, two states X and Y are **conditionally independent** given a third event Z precisely if the occurrence or non-occurrence of X and Y are **independent** events in their conditional probability distribution given Z.

In other words,

$P(X \cap Y | Z) = P(X|Z)\ P(Y|Z)$

Or equivalently, $P(X|Y \cap Z) = P(X|Z)$

Formal Statistics Definition

A Markov chain is a sequence of **random variables** X_1, X_2, X_3, ... with the **Markov property**, namely that, given the present state, the future and past states are independent.

Formally, $P(X_{n+1}=x|X_n=x_n,\ldots,X_1=x_1)=P(X_{n+1}=x|X_n=x_n)$

The possible values of X_i form a **countable set** *S* called the **state space** of the chain.

(We will be restricting ourselves to finite sets.)

Markov chains are often described by a **directed graph**, where the edges are labeled by the probabilities of going from one state to the other states.

Introduction to Probability

We describe a Markov chain as follows:

We have a set of **states**, S = {s1 , s2 , . . . , sr }.

The process starts in one of these states and moves successively from one state to another. Each move is called a **step**.

If the chain is currently in state s_i , then it moves to state s_j at the next step with a probability denoted by p_{ij} , and this probability does not depend upon which states the chain was in before the current state.

The probabilities p_{ij} are called **transition probabilities**.

The process can remain in the state it is in, and this occurs with probability p_{ii} .

An initial probability distribution, defined on S, specifies the starting state. Usually this is done by specifying a particular state as the starting state.

Land of Oz

According to Kemeny, Snell, and Thompson[2], the Land of Oz is blessed by many things, but not by good weather.

They never have two nice days in a row.

If they have a nice day, they are just as likely to have snow as rain the next day.

If they have snow or rain, they have an even chance of having the same the next day.

If there is change from snow or rain, only half of the time is this a change to a nice day.

Example Continued

With this information we form a Markov chain as follows.

We take as states the kinds of weather R, N, and S.

From the above information we determine the transition probabilities. These are most conveniently represented in a square array as

$$P = \begin{array}{c} \\ R \\ N \\ S \end{array} \begin{array}{c} \begin{array}{ccc} R & N & S \end{array} \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$

Hidden Markov Models

A **hidden Markov model (HMM)** is a **statistical model** in which the system being modeled is assumed to be a **Markov process** with unknown parameters, and the challenge is to determine the **hidden** parameters from the **observable** parameters.

A HMM can be considered as the simplest **dynamic Bayesian network**. (DBN means the arcs are directed. If the arcs aren't directed you have Markov Random Fields (MRFs) or Markov networks)

In a **regular Markov model**, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.

In a **hidden Markov model**, the state is not directly visible, but variables influenced by the state are visible.

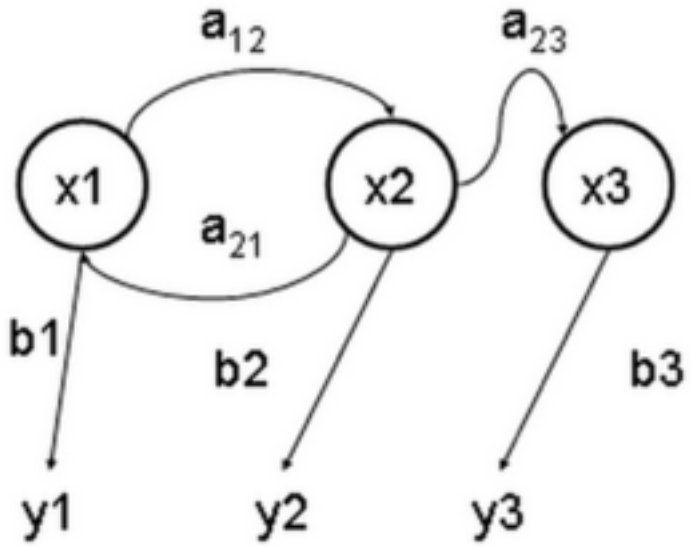
HMMs Continued

Each state has a probability distribution over the possible output tokens.

Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.

Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following and **bioinformatics**.

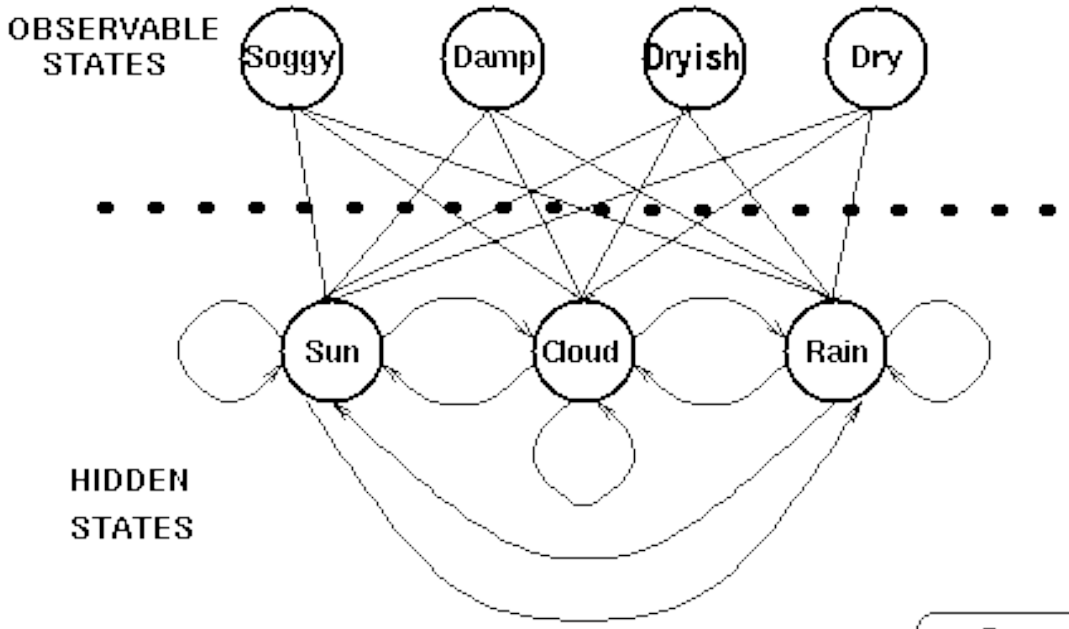
Hidden Markov Model Graph

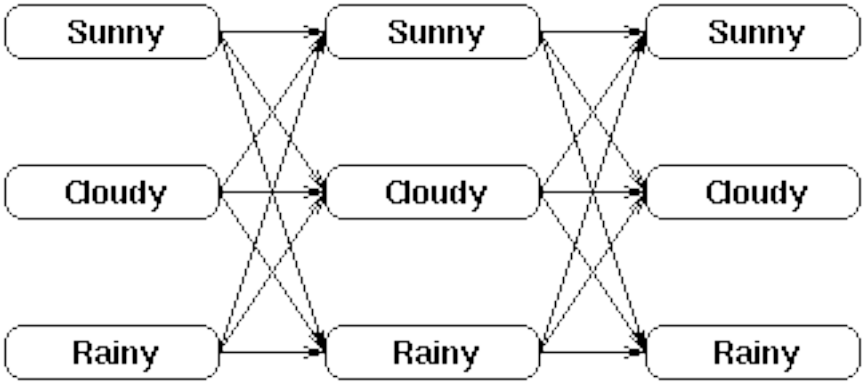


Why Hidden Markov Models

- http://www.cs.umd.edu/~djacobs/CMSC828/ApplicationsHMMs.pdf

Trellis Layout





Probability of HMM

The probability of observing a sequence $Y = y(0),y(1),...,y(L-1)$ of length L is given by:

$P(Y) = \Sigma_X P(Y|X) P(X)$

where the sum runs over all possible hidden node sequences $X = x(0),x(1),...,x(L-1)$.

Brute force calculation of $P(Y)$ is intractable for most real-life problems, as the number of possible hidden node sequences is going to be extremely high.

The calculation can however be sped up enormously using an algorithm called the forward procedure.

HMM CS Definition

A variant of a ***finite state machine*** having a set of **states**, Q, an output **alphabet**, O, **transition probabilities**, A, **output probabilities**, B, and **initial state probabilities**, Π.

The current state is not observable. Instead, each state produces an output with a certain probability (B).

Usually the states, Q, and outputs, O, are understood, so an HMM is said to be a triple, (A, B, Π).

HMM Formal CS definition

$A = \{a_{ij} = P(q_j \text{ at } t+1 \mid q_i \text{ at } t)\}$, where $P(x \mid y)$ is the conditional probability of x given y, $t \geq 1$ is time, and $q_i \in Q$.

Informally, A is the probability that the next state is q_j given that the current state is q_i.

$B = \{b_{ik} = P(o_k \mid q_i)\}$, where $o_k \in O$.

Informally, B is the probability that the output is o_k given that the current state is q_i.

$\Pi = \{p_i = P(q_i \text{ at } t=1)\}$.

HMM Formal Statistics Definition

States: A set of states $S=s_1,\ldots,s_n$

Transition probabilities: $A= a_{1,1},a_{1,2},\ldots,a_{n,n}$ Each $a_{i,j}$ represents the probability of transitioning from state s_i to s_j.

Emission probabilities: A set B of functions of the form $b_i(o_t)$ which is the probability of observation o_t being emitted by s_i

Initial state distribution: π_i is the probability that s_i is a start state

Hidden Markov Models
Summary

Frequently, patterns do not appear in isolation but as part of a series in time - this progression can sometimes be used to assist in their recognition.

Assumptions are usually made about the time based process - a common assumption is that the process's state is dependent only on the preceding N states - then we have an order N Markov model.

The simplest case is N=1, first-order Markov Model.

Stop here

Uses for Hidden Markov Models

Various examples exists where the process states (patterns) are not directly observable, but are indirectly, and probabilistically, observable as another set of patterns - we can then define a hidden Markov model - these models have proved to be of great value in many current areas of research, notably speech recognition.

How to use HMMs

Such models of real processes pose three problems that are amenable to immediate attack; these are:

Evaluation : with what probability does a given model generate a given sequence of observations. The **forward algorithm** solves this problem efficiently.

Decoding : what sequence of hidden (underlying) states most probably generated a given sequence of observations. The **Viterbi algorithm** solves this problem efficiently.

Learning : what model most probably underlies a given sample of observation sequences - that is, what are the parameters of such a model. This problem may be solved by using the **forward-backward** algorithm (or **Baum Welch**)

Success of HMMs

HMMs have proved to be of great value in analysing real systems.

Their usual drawback is the over-simplification associated with the Markov assumption

that a state is dependent only on predecessors, and that this dependence is time independent.

3 Main Problems - CS style

There are three **canonical** problems associated with HMMs:·

- **Evaluation**: Given the parameters of the model, compute the probability of a particular output sequence. This problem is solved by the **forward algorithm**.
- **Decoding**: Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This problem is solved by the **Viterbi algorithm**.
- **Learning**: Given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. In other words, train the parameters of the HMM given a dataset of sequences. This problem is solved by the **Baum-Welch algorithm**, **forward-backward algorithm**, **EM algorithm**.

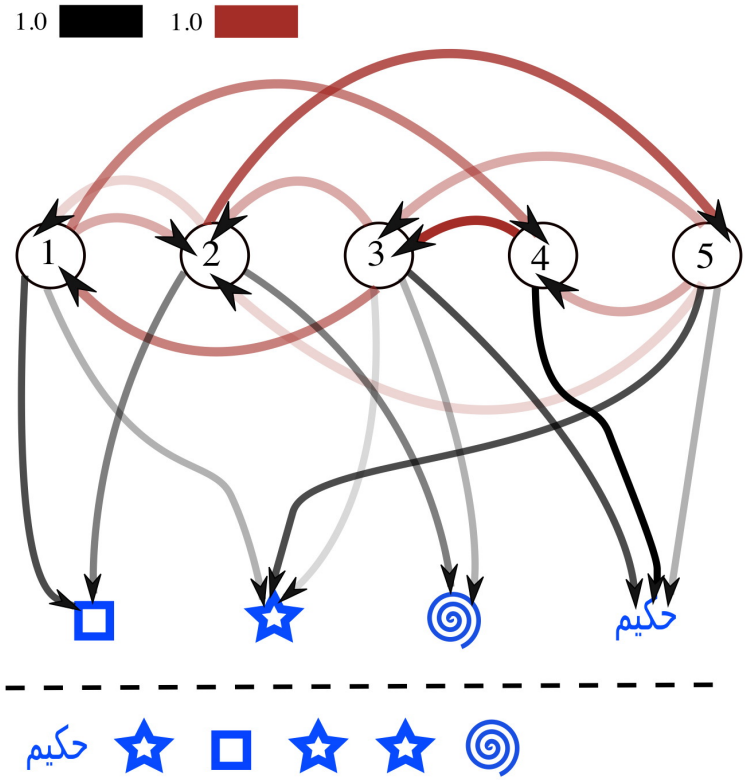
References

- CMSC 723: Introduction to Computational Linguistics - **University of Maryland Institute for Advanced Computer Studies** - www.umiacs.umd.edu/~christof/courses/ **cmsc723**-fall04/lecture-notes/Lecture5-hmm.ppt
- Leeds - http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels
- Wikipedia - http://en.wikipedia.org/wiki/Markov_chain
- Nist - http://www.nist.gov/dads/HTML/markovchain.html
- Introduction to Probability - Charles M Grimstead J Laurie Snell - http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf
- 2 J. G. Kemeny, J. L. Snell, G. L. Thompson, Introduction to Finite Mathematics, 3rd ed. (Englewood Cliffs, NJ: Prentice-Hall, 1974).
- **www.ncbioportal.org/ training/mod3/hiddenmark.html**
- http://www.ncbioportal.org/training/mod3/hmmer.html

HMM 2

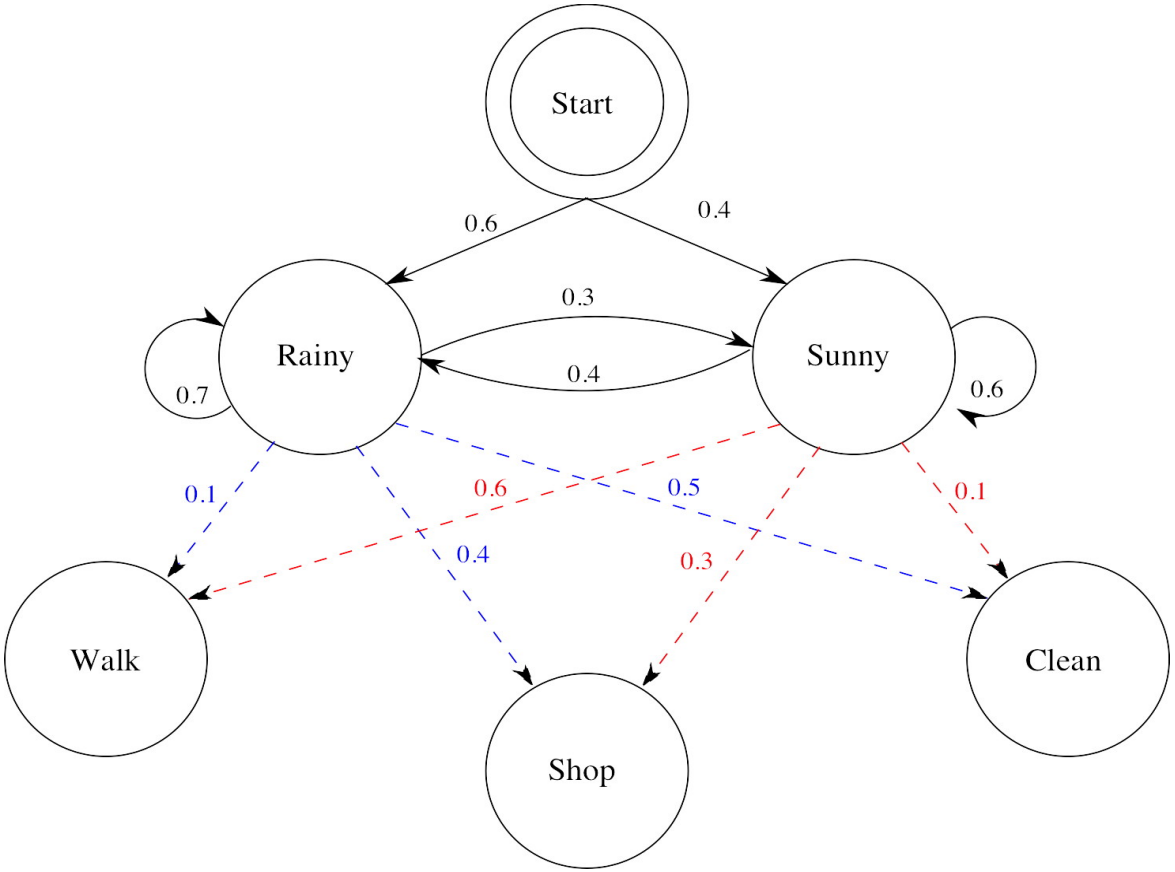
HMM Example

http://en.wikipedia.org/wiki/Hidden_Markov_model

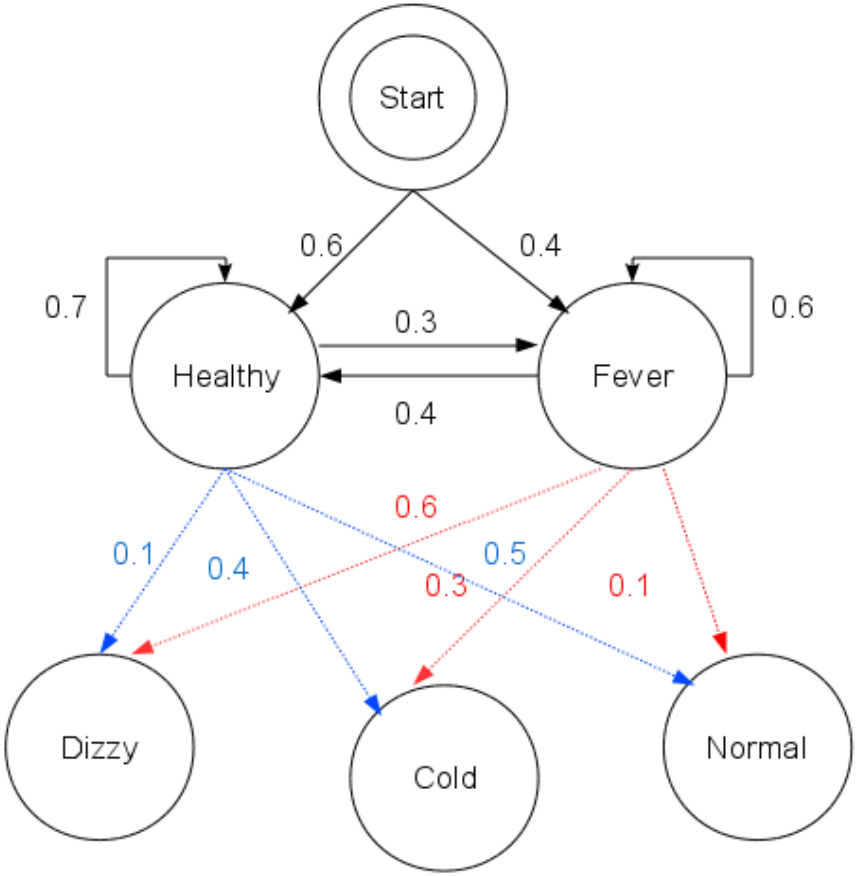


Another HMM Example

http://en.wikipedia.org/wiki/Hidden_Markov_model



Yet Another HMM example



Evaluation - Forward Algorithm

The forward algorithm, in the context of a hidden Markov model, is used to calculate a 'belief state': the probability of a state at a certain time, given the history of evidence.

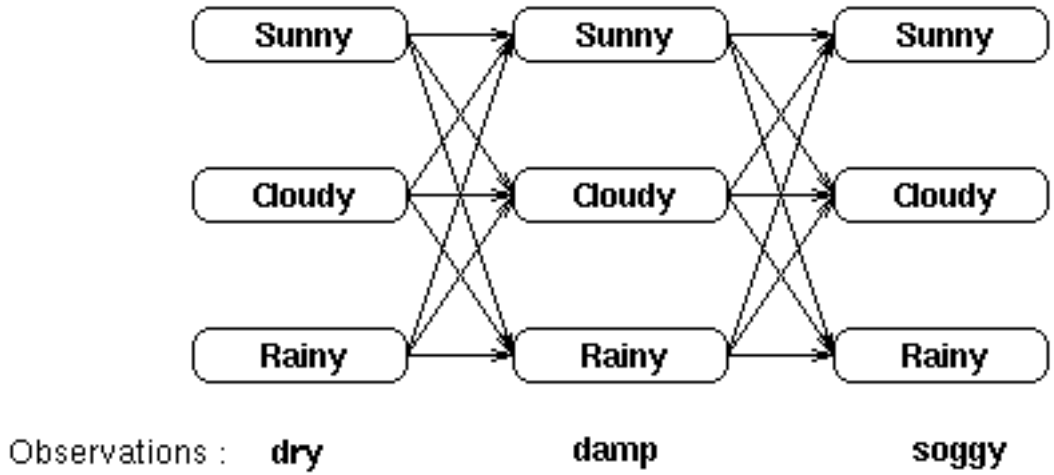
The forward algorithm is closely related to, but distinct from, the Viterbi algorithm.

Evaluation - Finding the probability of an observed sequence

Exhaustive search for solution

- We want to find the probability of an observed sequence given an HMM - that is, the parameters (π,A,B) are known.
- Consider the weather example; we have a HMM describing the weather and its relation to the state of the seaweed, and we also have a sequence of seaweed observations.
- Suppose the observations for 3 consecutive days are (dry,damp,soggy) - on each of these days, the weather may have been sunny, cloudy or rainy.
- We can picture the observations and the possible hidden states as a trellis.

Trellis


Sunny
Sunny
Sunny
Cloudy
Cloudy
Cloudy
Rainy
Rainy
Rainy
Observations :
dry
damp
soggy

Trellis Description

Each column in the trellis shows the possible state of the weather and each state in one column is connected to each state in the adjacent columns.

Each of these state transitions has a probability provided by the state transition matrix.

Under each column is the observation at that time; the probability of this observation given any one of the above states is provided by the confusion matrix.

It can be seen that one method of calculating the probability of the observed sequence would be to find each possible sequence of the hidden states, and sum these probabilities.

Brute Force Calculation

For the above example, there would be 3^3=27 possible different weather sequences, and so the probability is

$$P(\text{dry,damp,soggy} \mid \text{HMM}) =$$
$$P(\text{dry,damp,soggy} \mid \text{sunny,sunny,sunny}) +$$
$$P(\text{dry,damp,soggy} \mid \text{sunny,sunny ,cloudy}) +$$
$$P(\text{dry,damp,soggy} \mid \text{sunny,sunny ,rainy}) + \ldots.$$
$$P(\text{dry,damp,soggy} \mid \text{rainy,rainy ,rainy})$$

Calculating the probability in this manner is computationally expensive, particularly with large models or long sequences, but we can use the time invariance of the probabilities to reduce the complexity of the problem.

Problem 1: Probability of an Observation Sequence - Evaluation

What is $P(O \mid \lambda)$?

The probability of a observation sequence is the sum of the probabilities of all possible state sequences in the HMM.

Naïve computation is very expensive. Given T observations and N states, there are N^T possible state sequences.

Even small HMMs, e.g. T=10 and N=10, contain 10 billion different paths

Solution to this (and problem 2) is to use dynamic programming

Bugs

- Naive algorithm
  1. start a bug at state 0, time 0, holding value 0
  2. move each bug forward in time by making copies of it and incrementing the value of each copy by the probability of the transition and symbol emission
  3. go to 2 until all bugs have reached time T
  4. sum up values on all bugs

- Clever recursion
  - adds a step between 2 and 3 above which says at each node replace all the bugs at a state with a single bug carrying the sum of their values

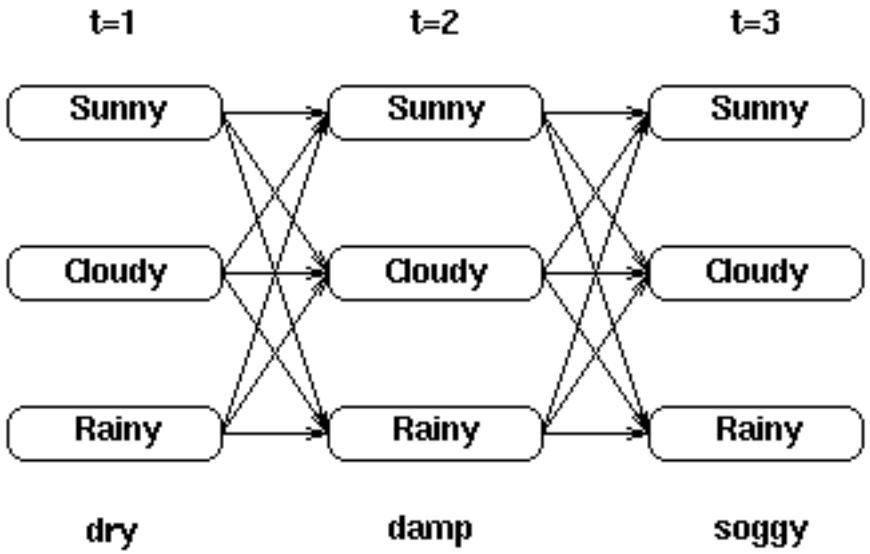
Reduction of complexity using recursion

- We will consider calculating the probability of observing a sequence recursively given a HMM.
- We will first define a partial probability, which is the probability of reaching an intermediate state in the trellis.
- We then show how these partial probabilities are calculated at times t=1 and t=n (> 1).
- Suppose throughout that the T-long observed sequence is

$$(Y_{k_1} , Y_{k_2} , \ldots\ldots , Y_{k_T})$$

Partial Probabilities

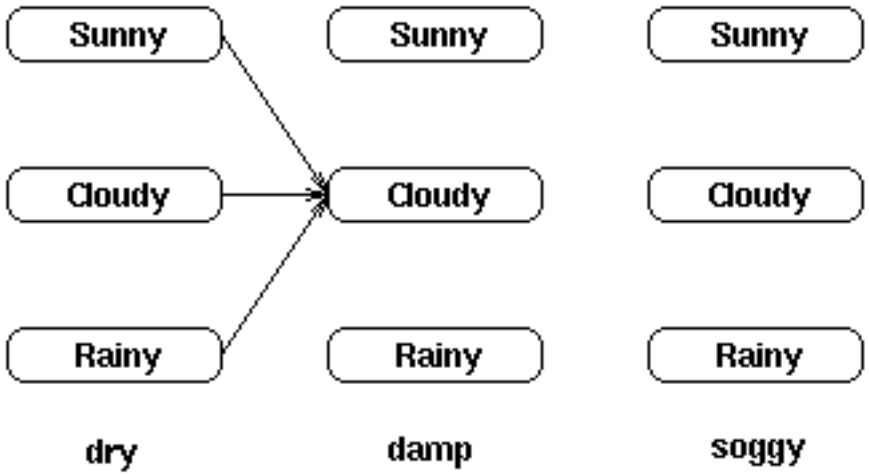
Consider the trellis below showing the states and first-order transitions for the observation sequence dry,damp,soggy;



Intermediate States

We can calculate the probability of reaching an intermediate state in the trellis as the sum of all possible paths to that state.

For example, the probability of it being cloudy at t=2 is calculated from the paths;



Partial Probabilities

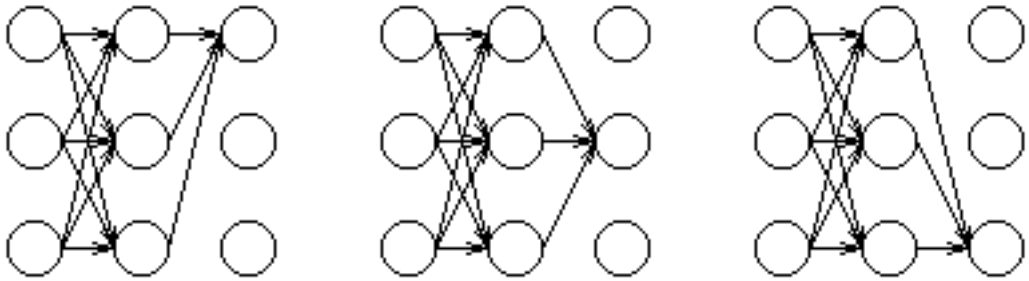
We denote the partial probability of state j at time t as t(j) - this partial probability is calculated as;

t(j)= P(observation | hidden state is j) x

P(all paths to state j at time t)

The partial probabilities for the final observation hold the probability of reaching those states going through all possible paths - e.g., for the previous trellis, the final partial probabilities are calculated from the following paths

Sum of Partial Probabilities



It follows that the sum of these final partial probabilities is the sum of all possible paths through the trellis, and hence is the probability of observing the sequence given the HMM.

Reduction of Computational Complexity

We can compare the computational complexity of calculating the probability of an observation sequence by exhaustive evaluation and by the recursive forward algorithm.

We have a sequence of T observations, O.

We also have a Hidden Markov Model, (π,A,B), with N hidden states.

Computational Complexity

An exhaustive evaluation would involve computing for all possible execution se $X_i = (X_{i_1}, X_{i_2}, \ldots, X_{i_T})$

the quant $\sum_X \pi(i_1) b_{i_1 k_1} \prod_{j=2}^{T} \alpha_{i_{j-1} i_j} b_{i_j k_j}$

which sums the probability of observing what we do - note that the complexity here is exponential in T.

- Conversely, using the forward algorithm we can exploit knowledge of the previous time step to compute information about a new one - accordingly, the complexity will only be linear in T.

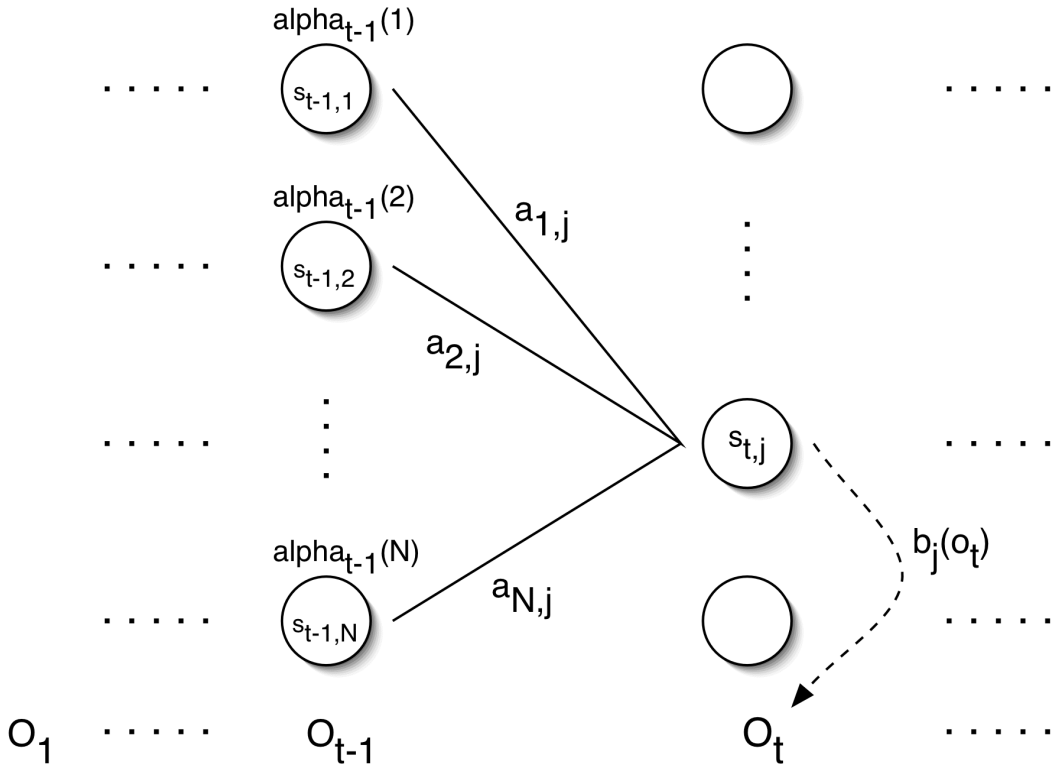
Forward Probabilities

What is the probability, given an HMM λ, that at time t the state is i and the partial observation $o_1, \ldots o_t$ has been generated?

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda)$$

Visual Forward

$$\alpha_t(i) = P(o_1 .. o_t, q_t = s_i \mid \lambda)$$



$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Forward Recursive Function

Initialization: $\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$

Induction:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right] b_j(o_t) \quad 2 \le t \le T, 1 \le j \le N$$

Termination:

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

Detailed Computational Complexity

In the naïve approach to solving problem 1 it takes on the order of $T*N^T$ computations

The forward algorithm takes on the order of N^2T computations

Backward Algorithm

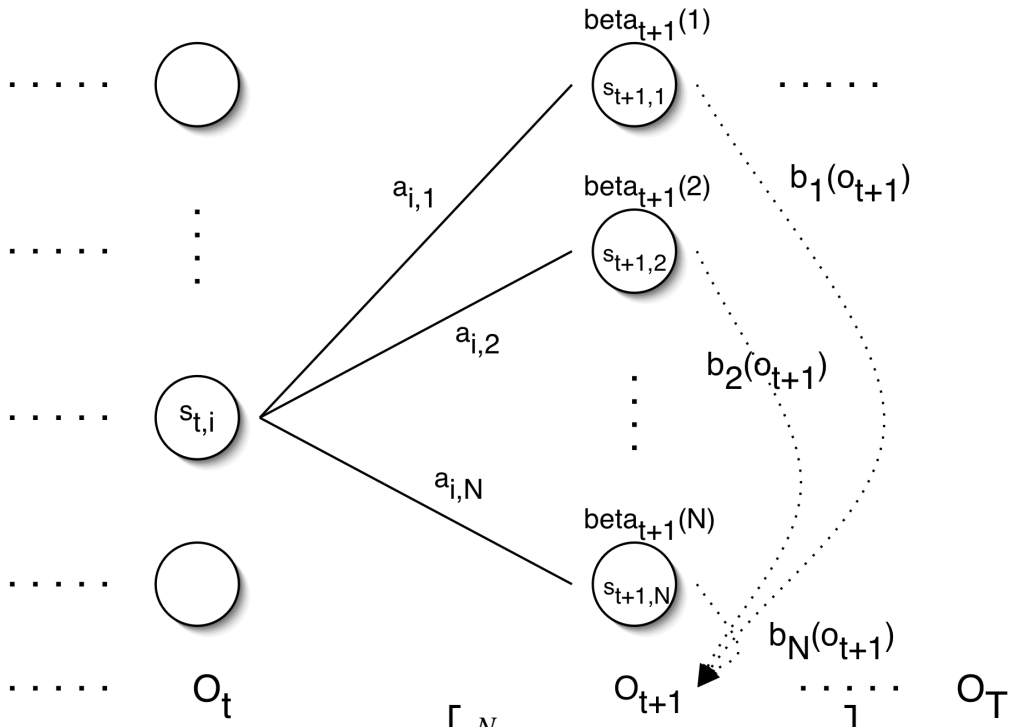
Analogous to the forward probability, just in the other direction

What is the probability that given an HMM and given the state at time t is i, the partial observation $o_{t+1} \ldots o_T$ is generated?

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$

Backward Visual

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$



$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right]$$

Backward Recursive Function

Initialization: $\beta_T(i) = 1, \quad 1 \le i \le N$

Induction:

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right] \quad t = T-1 \ldots 1, 1 \le i \le N$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i)$$

Comparing Forward and Backward

Notice that in the definitions above, the forward probability, $P(o_1 \ldots o_t, q_t=s_i \mid \lambda)$, is a **joint probability** whereas the backward probability, $P(o_{t+1},\ldots o_T \mid q_t=s_i,\lambda)$ is a **conditional probability**.

This somewhat asymmetric definition is deliberate since it allows the probability of state occupation to be determined by taking the product of the two probabilities:

$$\alpha_j(t)\beta_j(t) = P(O, q_t = s_j \mid \lambda)$$

We will talk about this more during the Backward-Forward or Baum-Welch algorithm.

Summary

- Our aim is to find the probability of a sequence of observations given a HMM - Pr(observations |λ).
- We reduce the complexity of calculating this probability by first calculating partial probabilities ('s). These represent the probability of getting to a particular state, s, at time t.
  - At time t = 1, the partial probabilities are calculated using the initial probabilities (from the vector) and Pr(observation|state) (from the confusion matrix);
  - At time t (> 1), the partial probabilites can be calculated using the partial probabilities at time t-1.
- This definition of the problem is recursive, and the probability of the observation sequence is found by calculating the partial probabilities at time t = 1, 2, ..., T, and adding all 's at t = T.
- Notice that computing the probability in this way is far less expensive than calculating the probabilities for all sequences and adding them.

Decoding

The solution to Problem 1 (Evaluation) gives us the sum of all paths through an HMM efficiently.

For Problem 2 (Decoding), we want to find the path with the highest probability.

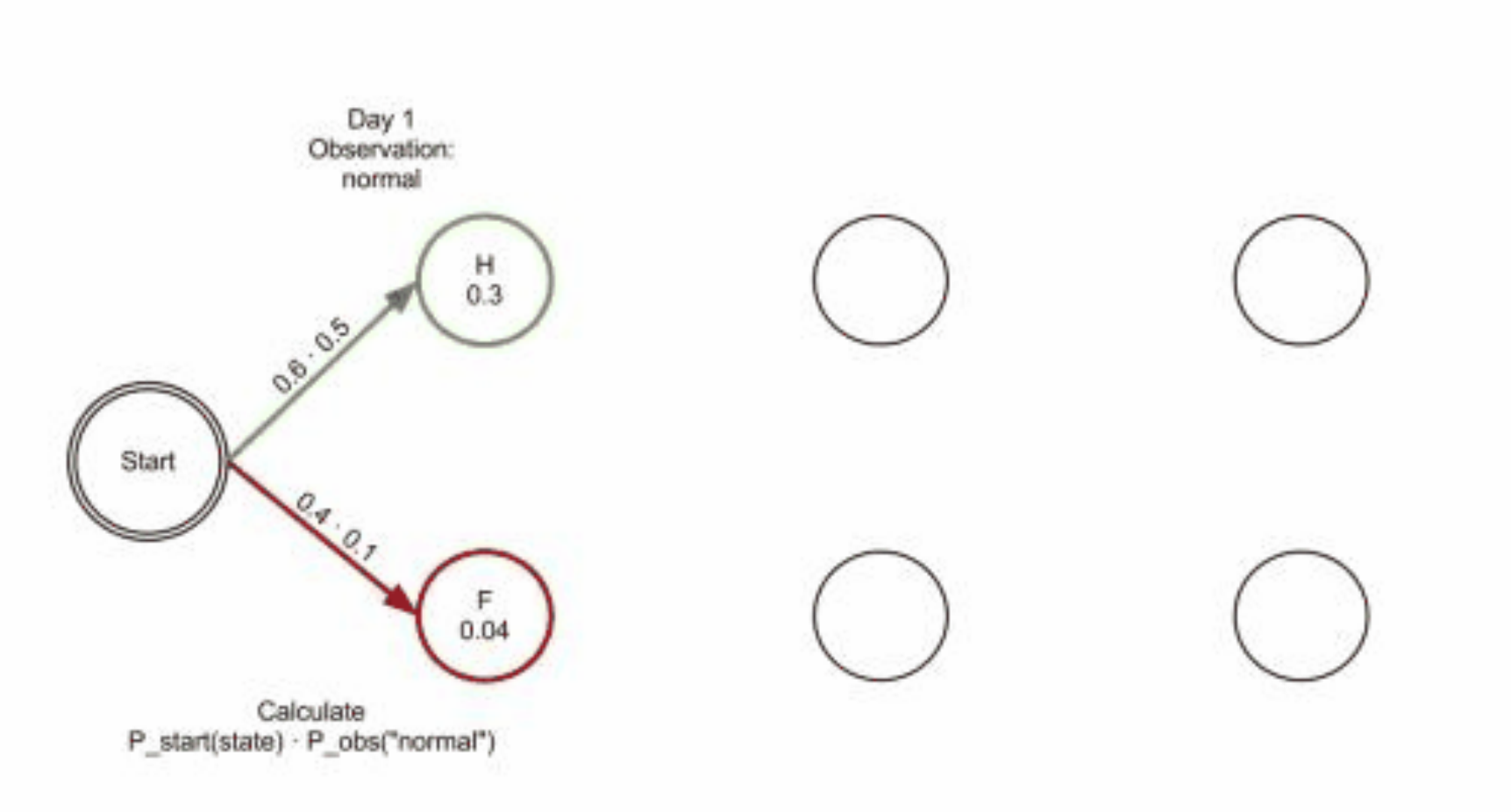
We want to find the state sequence $Q=q_1 \ldots q_T$, such that

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda)$$

Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models.

Viterbi Algorithm

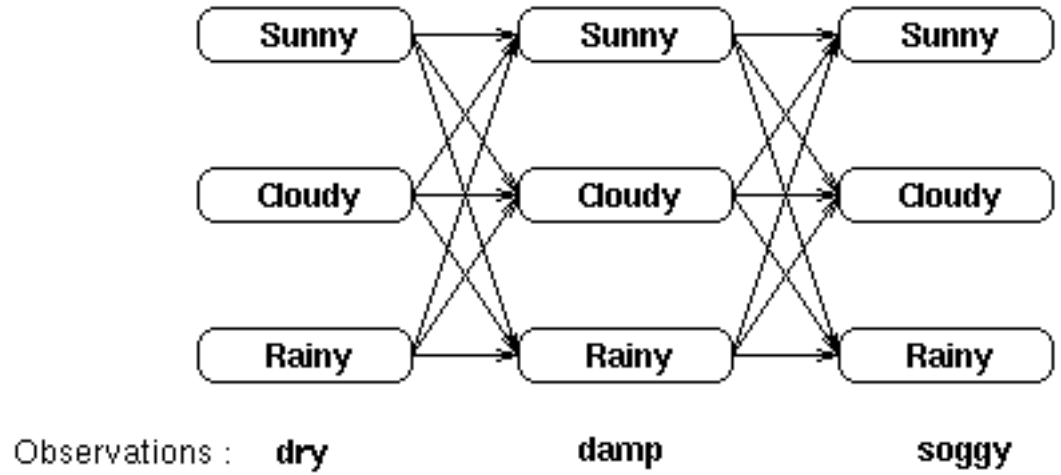


Most Probably Sequence of Hidden States

We often wish to take a particular HMM, and determine from an observation sequence the most likely sequence of underlying hidden states that might have generated it.

We can use a picture of the execution trellis to visualise the relationship between states and observations.

Trellis


Sunny
Sunny
Sunny
Cloudy
Cloudy
Cloudy
Rainy
Rainy
Rainy
Observations :
dry
damp
soggy

Maximising Probability

We can find the most probable sequence of hidden states by listing all possible sequences of hidden states and finding the probability of the observed sequence for each of the combinations.

The most probable sequence of hidden states is that combination that maximises

P(observed sequence | hidden state combination).

Brute Force Solution

For example, for the observation sequence in the trellis shown, the most probable sequence of hidden states is the sequence that maximises :

P(dry,damp,soggy | sunny,sunny,sunny),

P(dry,damp,soggy | sunny,sunny,cloudy),

P(dry,damp,soggy | sunny,sunny,rainy), ...

P(dry,damp,soggy | rainy,rainy,rainy)

This approach is viable, but to find the most probable sequence by exhaustively calculating each combination is computationally expensive.

As with the forward algorithm, we can use the time invariance of the probabilities to reduce the complexity of the calculation.

Reducing complexity using recursion

We will consider recursively finding the most probable sequence of hidden states given an observation sequence and a HMM.

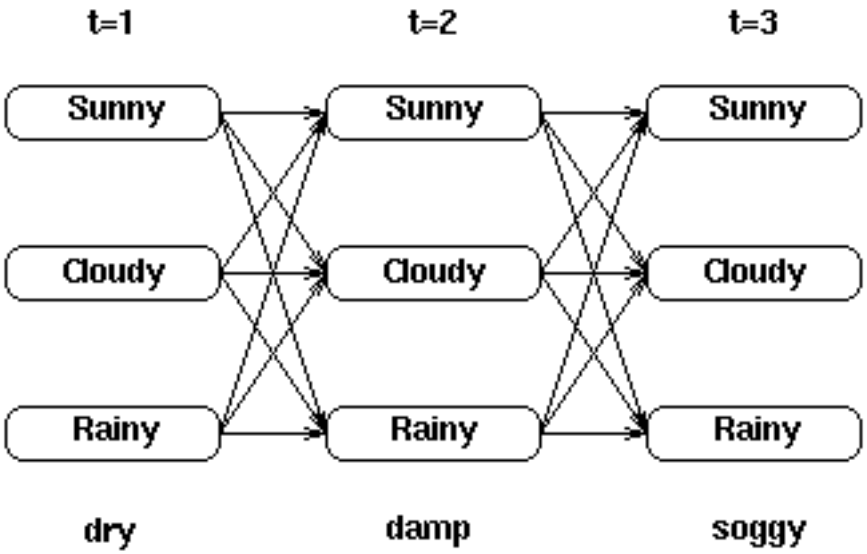
We will first define the partial probability, δ , which is the probability of reaching a particular intermediate state in the trellis.

We then show how these partial probabilities are calculated at t=1 and at t=n (> 1).

These partial probabilities differ from those calculated in the forward algorithm since they represent the probability of the most probable path to a state at time t, and not a total.

Partial Probabilities

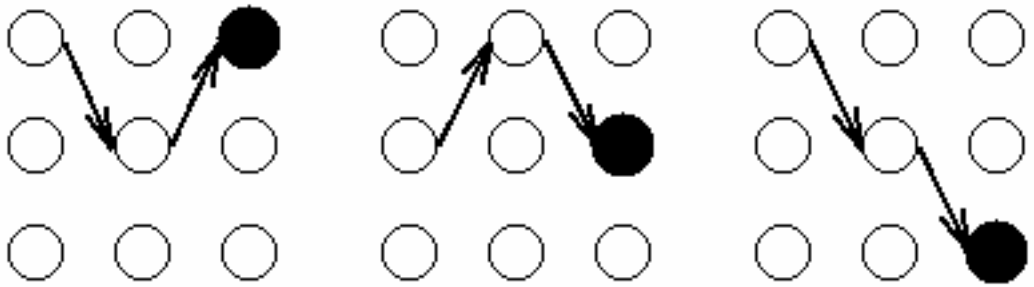
Consider the trellis below showing the states and first order transitions for the observation sequence dry,damp,soggy;



Most Probable Path

For each intermediate and terminating state in the trellis there is a most probable path to that state.

So, for example, each of the three states at t=3 will have a most probable path to it, perhaps like this;



Partial Best Paths

We will call these paths partial best paths.

Each of these partial best paths has an associated probability, the partial probability or δ.

Unlike the partial probabilities in the forward algorithm, δ is the probablity of the one (most probable) path to the state.

Thus $\delta(i,t)$ is the maximum probability of all sequences ending at state i at time t, and the partial best path is the sequence which achieves this maximal probability.

Such a probability (and partial path) exists for each possible value of i and t.

In particular, each state at time t = T will have a partial probability and a partial best path. We find the overall best path by choosing the state with the maximum partial probability and choosing its partial best path.

Advantages of Virterbi

Using the Viterbi algorithm to decode an observation sequence carries two important advantages:

1. There is a reduction in computational complexity by using the recursion - this argument is exactly analogous to that used in justifying the forward algorithm.

Advantages continued

2. The Viterbi algorithm has the very useful property of providing the best interpretation given the entire context of the observations.

The Viterbi algorithm will look at the whole sequence before deciding on the most likely final state, and then `backtracking' through the f pointers to indicate how it might have arisen.

This is very useful in `reading through' isolated noise garbles, which are very common in live data.

Viterbi Algorithm

Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum

Forward:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Viterbi Recursion:

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Viterbi Recursive Function

$$\delta_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$$

Initialization:

Induction:

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}\right] b_j(o_t)$$

$$\psi_t(j) = \left[\arg\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}\right] \quad 2 \le t \le T, 1 \le j \le N$$

Termination: $p^* = \max_{1 \le i \le N} \delta_T(i)$ $\qquad q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$

Read out path:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1,\ldots,1$$

Viterbi Summary

The Viterbi algorithm provides a computationally efficient way of analysing observations of HMMs to recapture the most likely underlying state sequence.

It exploits recursion to reduce computational load, and uses the context of the entire sequence to make judgements, thereby allowing good analysis of noise.

In use, the algorithm proceeds through an execution trellis calculating a partial probability for each cell, together with a back-pointer indicating how that cell could most probably be reached.

On completion, the most likely final state is taken as correct, and the path to it traced back to t=1 via the back pointers.

Baum-Welch Algorithm

The Baum–Welch algorithm is used to find the unknown parameters of a hidden Markov model (HMM). It makes use of the forward-backward algorithm and is named for Leonard E. Baum and Lloyd R. Welch.

The Baum–Welch algorithm uses the well known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors.

Purpose of Learning

The `useful' problems associated with HMMs are those of evaluation and decoding - they permit either a measurement of a model's relative applicability, or an estimate of what the underlying model is doing (what `really happened').

It can be seen that they both depend upon foreknowledge of the HMM parameters - the state transition matrix, the observation matrix, and the vector.

There are, however, many circumstances in practical problems where these are not directly measurable, and have to be estimated - this is the learning problem.

Learning

Up to now we've assumed that we know the underlying model $\lambda = (A, B, \pi)$

Often these parameters are estimated on annotated training data, which has two drawbacks:

- Annotation is difficult and/or expensive
- Training data is different from the current data

We want to maximize the parameters with respect to the current data, i.e., we're looking for a model λ', such that

$$\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$$

How it Works

The forward-backward algorithm permits this estimate to be made on the basis of a sequence of observations known to come from a given set, that represents a known hidden set following a Markov model.

An example may be a large speech processing database, where the underlying speech may be modeled by a Markov process based on known phonemes, and the observations may be modeled as recognisable states (perhaps via some vector quantisation), but there will be no (straightforward) way of deriving empirically the HMM parameters.

A Rose by any other name…

Forward-Backward algorithm is also called

Baum-Welch algorithm which is a type of

EM algorithm

Forward-Backward Intuition

The forward-backward algorithm is not unduly hard to comprehend, but is more complex in nature than the forward algorithm and the Viterbi algorithm.

The algorithm proceeds by making an initial guess of the parameters (which may well be entirely wrong) and then refining it by assessing its worth, and attempting to reduce the errors it provokes when fitted to the given data.

In this sense, it is performing a form of gradient descent, looking for a minimum of an error measure.

Basis of Forward-Backward

It derives its name from the fact that, for each state in an execution trellis, it computes the `forward' probability of arriving at that state (given the current model approximation) and the `backward' probability of generating the final state of the model, again given the current approximation.

Both of these may be computed advantageously by exploiting recursion, much as we have seen already.

Adjustments may be made to the approximated HMM parameters to improve these intermediate probabilities, and these adjustments form the basis of the algorithm iterations.

Baum-Welch

Unfortunately, there is no known way to analytically find a global maximum, i.e., a model $\hat{\lambda}$, such that $\hat{\lambda} = \arg\max_{\lambda} P(O \mid \lambda)$

But it is possible to find a local maximum

Given an initial model λ, we can always find a model λ', such that $P(O \mid \lambda') \geq P(O \mid \lambda)$

Parameter Re-estimation

Use the forward-backward (or Baum-Welch) algorithm, which is a hill-climbing algorithm

Using an initial parameter instantiation, the forward-backward algorithm iteratively re-estimates the parameters and improves the probability that given observation are generated by the new parameters

What needs to be Re-estimated?

Three parameters need to be re-estimated:

- Initial state distribution: π_i
- Transition probabilities: $a_{i,j}$
- Emission probabilities: $b_i(o_t)$

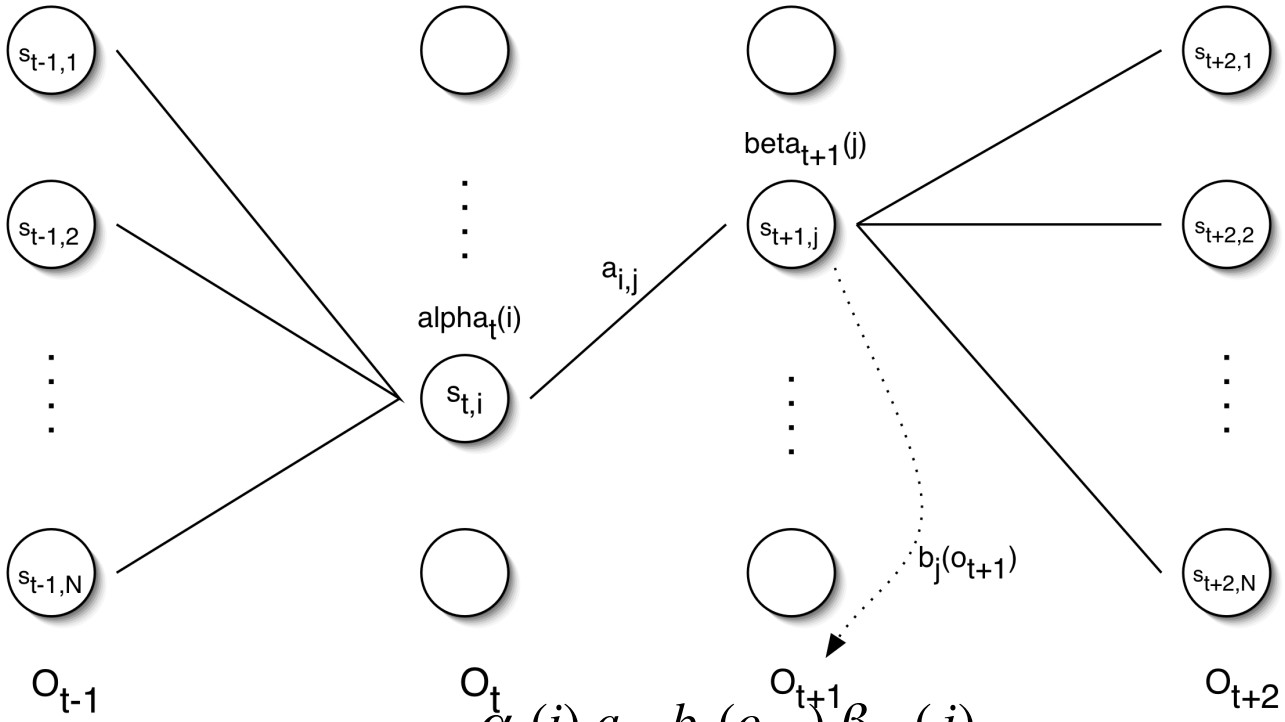
Re-estimating Transition Probabilites

What's the probability of being in state s_i at time t and going to state s_j, given the current model and parameters?

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

Visual Baum-Welch

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$



$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{i,j}\, b_j(o_{t+1})\, \beta_{t+1}(j)}$$

Transition re-estimation

The intuition behind the re-estimation equation for transition probabilities is

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i}$$

Formally:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j'=1}^{N} \xi_t(i,j')}$$

State Probability

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

As the probability of being in state s_i, given the complete observation O

We can say: $$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Review of Probabilities

$$\alpha_t(i)$$

Forward probability:

The probability of being in state s_i, given the partial observation $o_1,\ldots,o_t$

$$\beta_t(i)$$

Backward probability:

The probability of being in state s_i, given the partial observation $o_{t+1},\ldots,o_T$

$$\xi_t(i,j)$$

Transition probability:

The probability of going from state s_i, to state s_j, given the complete observation $o_1,\ldots,o_T$

$$\gamma_t(i)$$

State probability:

The probability of being in state s_i, given the complete observation $o_1,\ldots,o_T$

Initial State Re-estimation

π_i

Initial state distribution: is the probability that s_i is a start state

Re-estimation is easy:

$\hat{\pi}_i$ = expected number of times in state s_i at time 1

Formally:

$$\hat{\pi}_i = \gamma_1(i)$$

Emission Re-estimations

Emission probabilities are re-estimated as

$$b_i(k) = \frac{\text{expected number of times in state } s_i \text{ and observe symbol } v_k}{\text{expected number of times in state } s_i}$$

Formally:

$$\hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\,\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Where

$$\delta(o_t, v_k) = 1, \text{ if } o_t = v_k, \text{ and 0 otherwise}$$

NOTE: that δ here is the Kronecker delta function and is not related to the δ in the discussion of the Viterbi algorithm!!

The Updated Model

$$\lambda = (A, B, \pi)$$

Coming from $\lambda = (A, B, \pi)$ we get to $\lambda' = (\hat{A}, \hat{B}, \hat{\pi})$ by the following update rules:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\, \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \hat{\pi}_i = \gamma_1(i)$$

Product of two Probabilities

Notice that in the definitions above, the forward probability is a joint probability whereas the backward probability is a conditional probability.

This somewhat asymmetric definition is deliberate since it allows the probability of state occupation to be determined by taking the product of the two probabilities.

From the definitions,

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda)$$

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$

$$\alpha_j(t)\beta_j(t) = P(O, q_t = s_i \mid \lambda)$$

What is EM

The forward-backward algorithm is an instance of the more general EM algorithm

- The E Step: Compute the forward and backward probabilities for a give model
- The M Step: Re-estimate the model parameters

These two steps are repeated as necessary.

EM Algorithm

The EM algorithm [ALR77, RW84, GJ95, JJ94, Bis95, Wu83] is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set when the data is incomplete or has missing values.

There are two main applications of the EM algorithm.

- The first occurs when the data has missing values, due to problems with or limitations of the observation process.
- The second occurs when optimizing the likelihood function is analytically intractable but when the likelihood function can be simplified by assuming the existence of and values for additional but *missing* (or *hidden*) parameters. The latter application is more common in the computational pattern recognition community.

REFERENCES

www.cs.toronto.edu/~roweis/notes/hmm.ps.gz

http://www.ee.columbia.edu/ln/LabROSA/doc/HTKBook21/HTKBook.html

http://books.google.co.nz/books?id=BFnkm-FpBAUC

Start from this slide

Monte Carlo Stuff

Computer Science 760

Patricia J Riddle

Monte Carlo Markov Chain (MCMC)

- Monte Carlo – sampling
- Markov Chain – finite automata with arrows and probabilities

MCMC

- Desired distribution as its equilibrium distribution

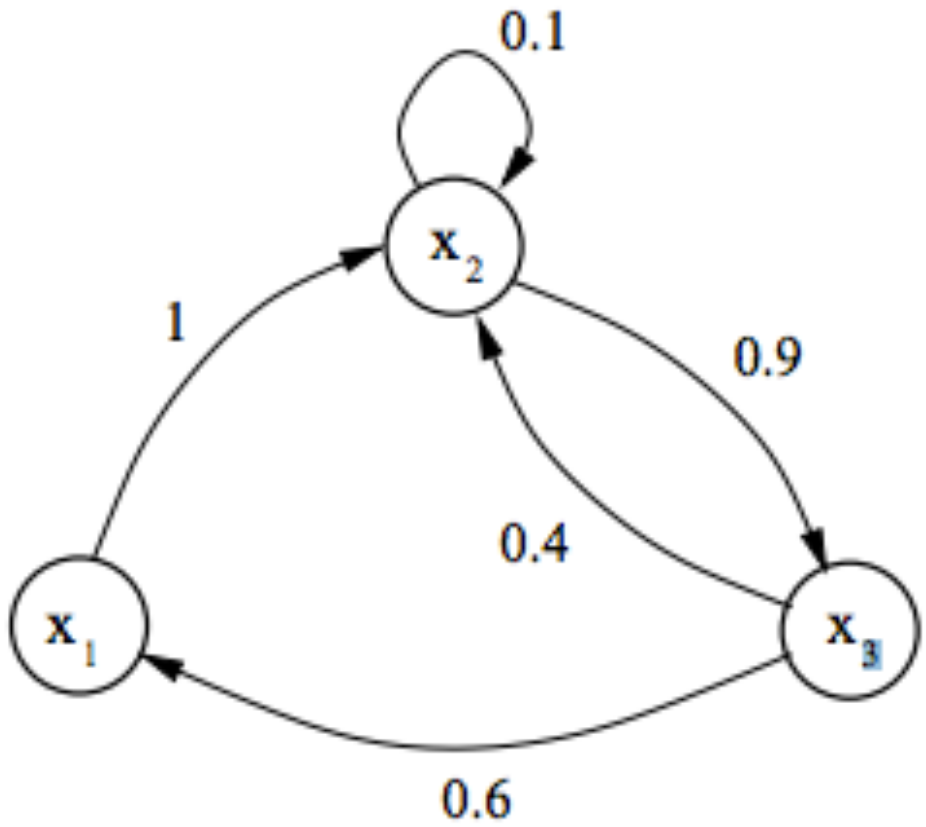
- State of chain after a large number of steps is a sample of the desired distribution

MCMC

- Quality of the sample improves as a function of the number of steps
- How many steps are needed to converge to a stationary distribution with an acceptable error?
- Only approximate – always some residual effect of the starting position

MCMC Example

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$



MCMC Example 2

- No matter what the initial distribution,
- After several iterations

- It will stabilise at p(x) = $(0.2, 0.4, 0.4)$

Invariant Distribution

If stochastic transition matrix obeys the following properties:

- Irreducibility: For any state, there is a positive probability of visiting all other states (the graph is connected)

- Aperiodicity: the chain should not get trapped in cycles

Application

- Calculating multi-dimensional integrals
  - Ensemble of walkers
  - At each point where the walker steps, the integrand value is counted towards the integral

Random Walk Algorithms

- Move around equilibrium in relatively small steps, no tendency for the steps to proceed in the same direction
- Will take a long time to explore all of the space
- Walker will often double back and cover ground already covered
- Metropolis-Hastings, Gibbs Sample, Slice sampling, Multiple-try Metropolis

Avoiding random walks

- Prevent walker from doubling back
- Harder to implement, faster convergence
- (only try to avoid random walks)
  - Successive over-relaxation
  - Hybrid Monte Carlo – Hamilton Monte Carlo, uses momentum and Hamilton dynamics to take larger steps

Monte Carlo Tree Search

- Used to estimate min-max probability

Problems with Go

Go

- Strongest programs weaker than amateur players

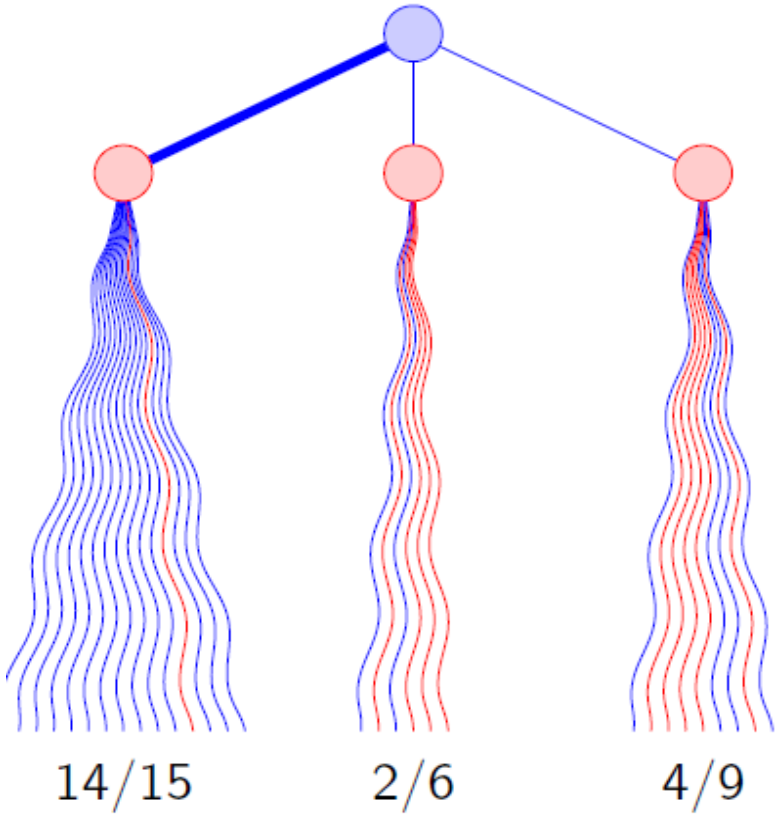
- No Good Static Evaluation Function for Go

Monte Carlo Tree Search

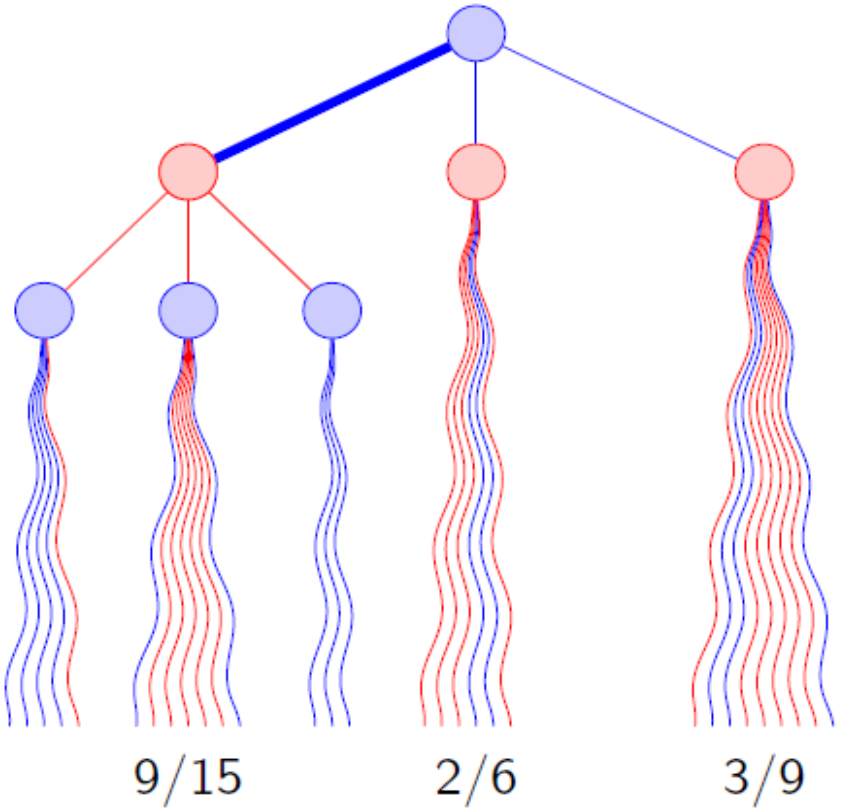
- Random playouts

Random playouts

store # wins and # plays



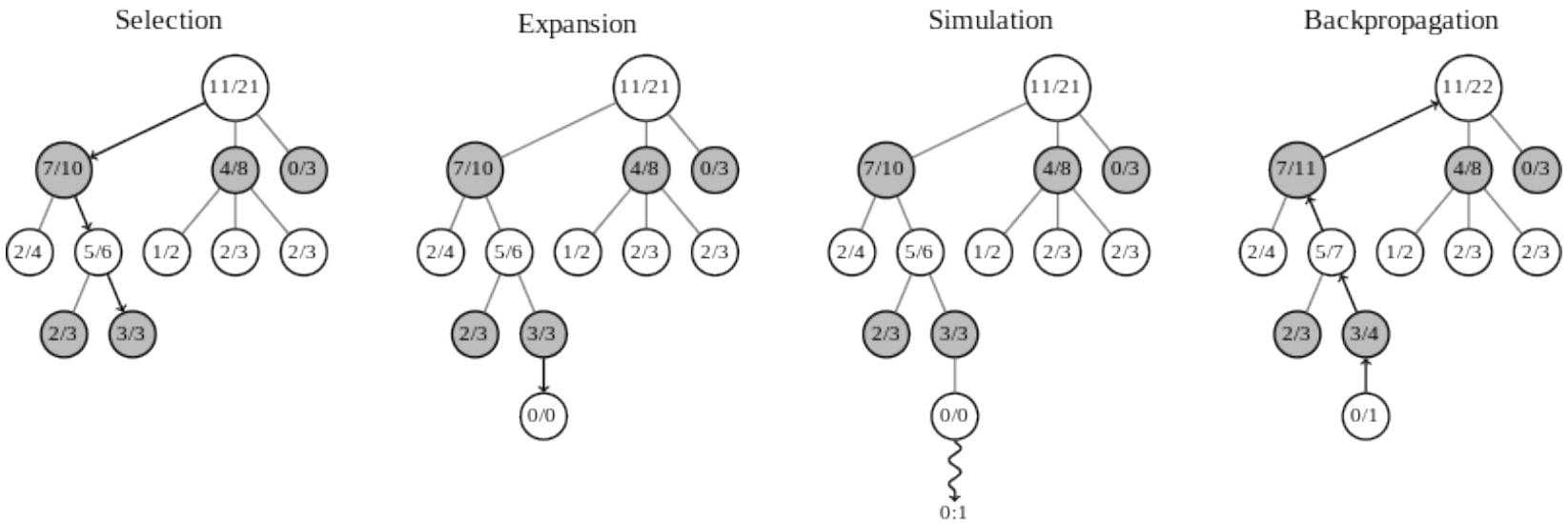
Make a move



Connect 4 Demo

http://beej.us/blog/data/monte-carlo-method-game-ai/

Monte Carlo Tree Search



Main Points of MCTS

- Evaluation of moves in MCTS converges to minimax evaluation
- No explicit evaluation function

Exploration vs Exploitation

- Converges to perfect play for (k to infinity)
  - If games with finite depth
  - Exploration and exploitation

UCT Formula
(Upper confidence bound)

- Exploration and exploitation

$$UCB_i = \frac{W_i}{N_i} + c\sqrt{\frac{\log t}{N_i}}$$

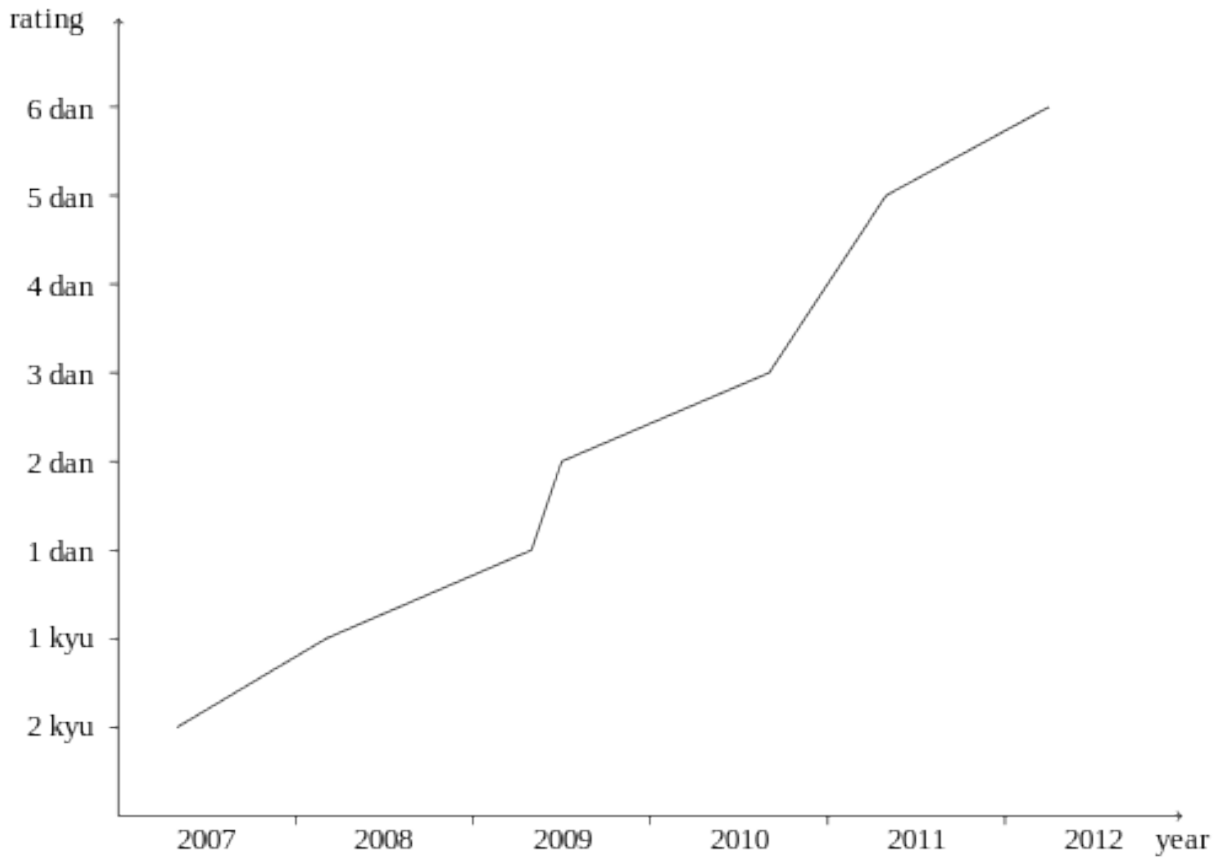
W_i wins (from move i)

N_i playouts (from move i)

c exploration parameter ($\sqrt{2}$ or chosen empirically)

t playouts (all moves)

Computer Go since MCTS



Improvements to MCTS by Jared Newman

- Using a DAG
  - Split aggregate

- Learning over multiple Runs

DAG aggregation-split

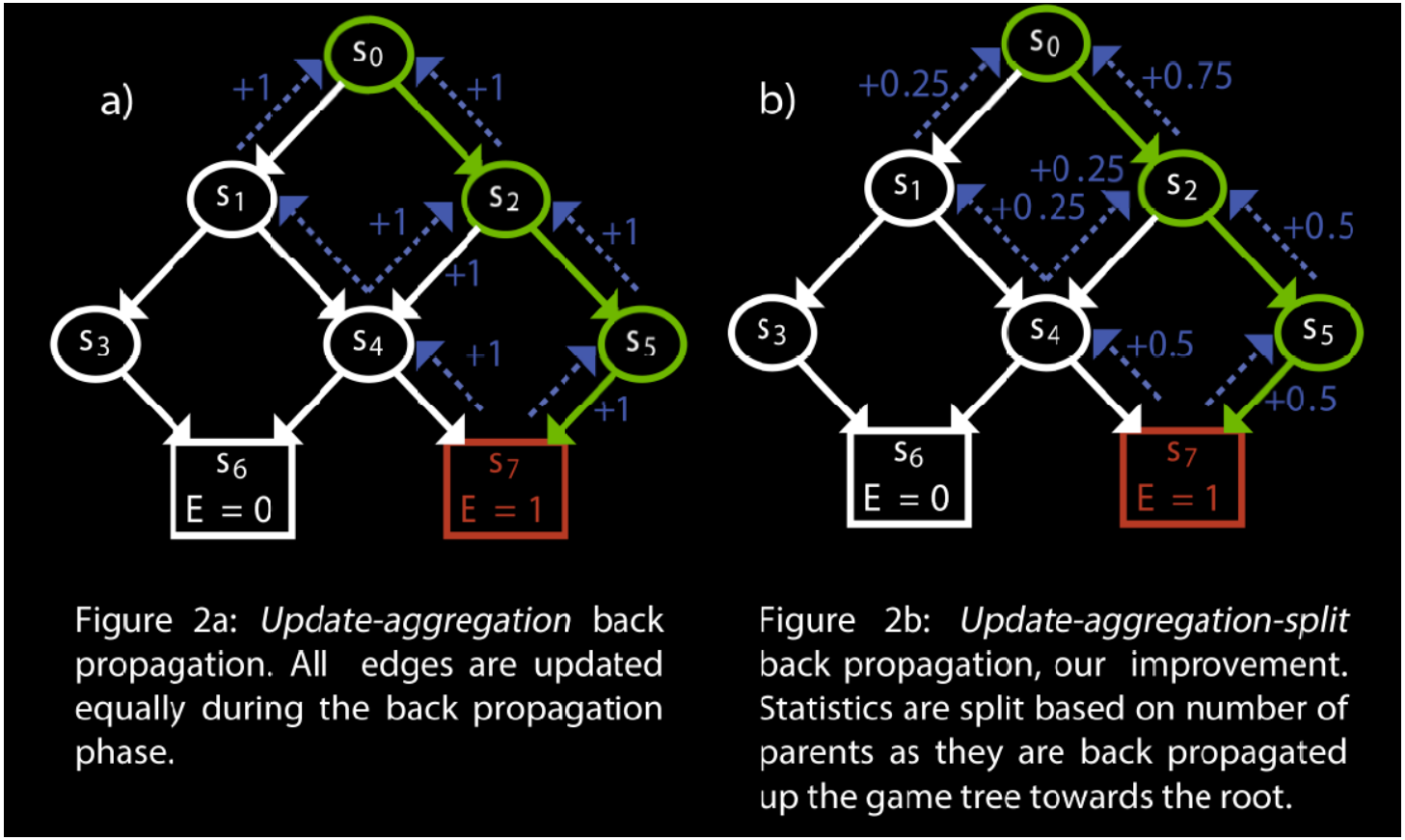
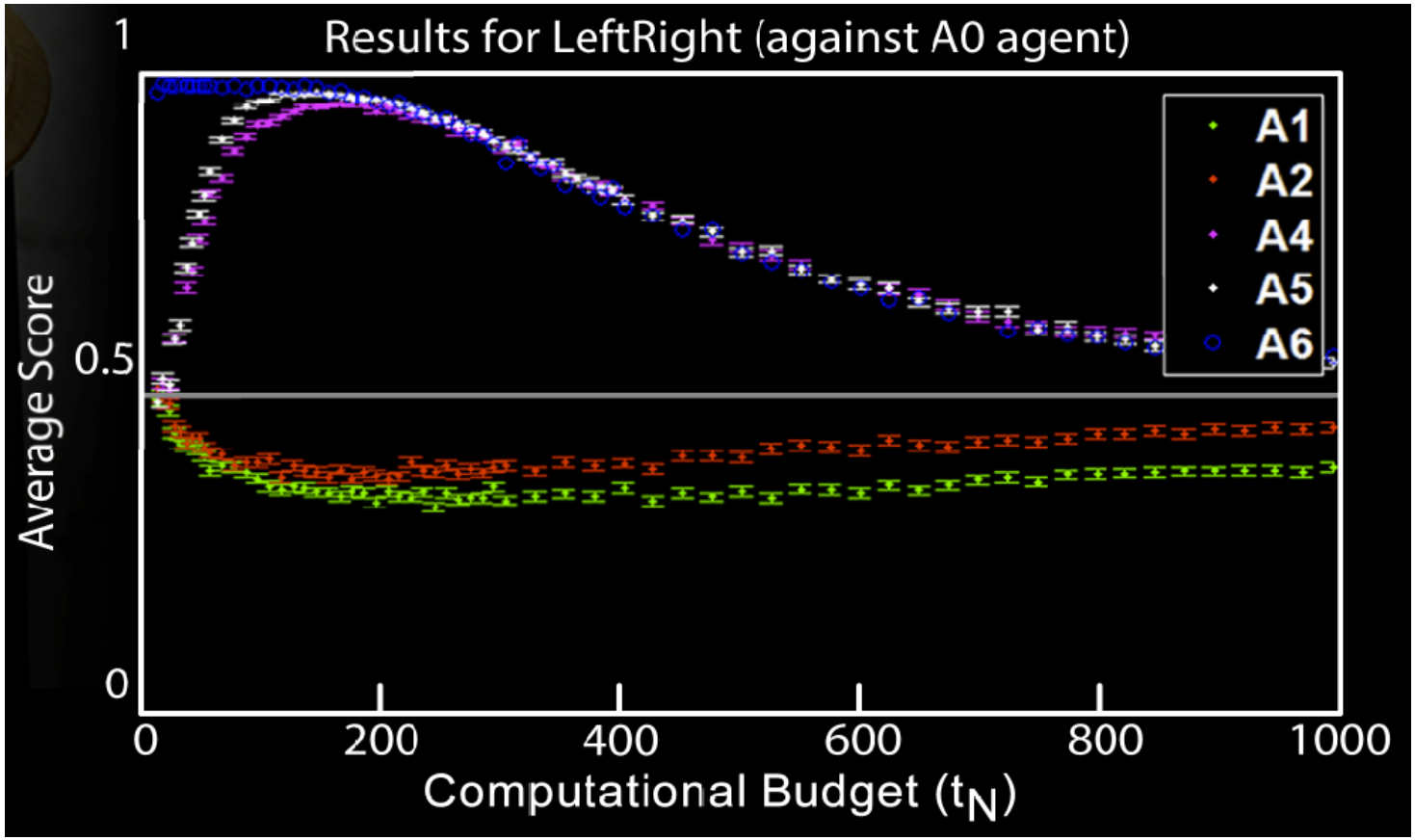


Figure 2a: *Update-aggregation* back propagation. All edges are updated equally during the back propagation phase.

Figure 2b: *Update-aggregation-split* back propagation, our improvement. Statistics are split based on number of parents as they are back propagated up the game tree towards the root.

Results with Learning



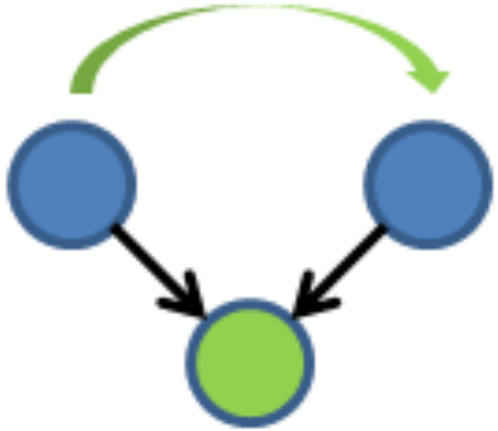
Markov Random Fields

A Markov random field is a set of random variables having a Markov property described by an undirected graph.

A Markov random field is similar to a Bayesian network in its representation of dependencies; the differences being that Bayesian networks are directed and acyclic, whereas Markov networks are undirected and may be cyclic.

Thus, a Markov network can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies); on the other hand, it can't represent certain dependencies that a Bayesian network can (such as induced dependencies).

Induced Dependencies



There are dependencies that an MRF cannot represent.

One of these is the induced dependency.

If the child node is observed and has multiple parents, in a Bayesian network (BN) information can flow between the parent nodes.

In an MRF, no information can flow across known nodes.

Stop again

Markov Random Fields

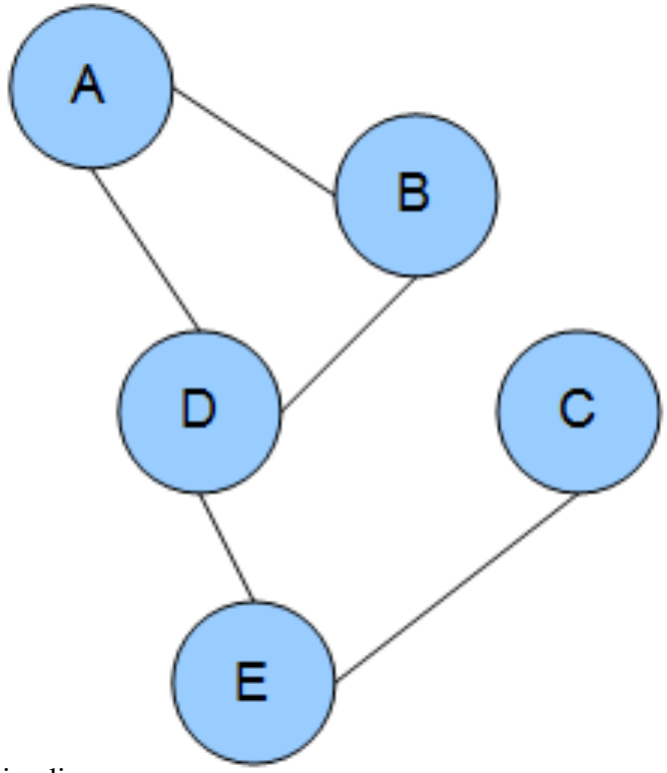
When the probability distribution is strictly positive, it is also referred to as a Gibbs random field.

In the domain of artificial intelligence, a Markov random field is used to model various low- to mid-level tasks in image processing and computer vision.[2]

For example, MRFs are used for image restoration, image completion, segmentation, image registration, texture synthesis, super-resolution, stereo matching and information retrieval.

Markov Random Field

http://en.wikipedia.org/wiki/Markov_random_field



"Markov random field example". Via Wikipedia – http://en.wikipedia.org/wiki/File:Markov_random_field_example.png#mediaviewer/File:Markov_random_field_example.png

Markov Random Field

Given an undirected graph $G = (V, E)$, a set of random variables $X = (X_v)_{v \in V}$ indexed by V form a Markov random field with respect to G if they satisfy the local Markov properties:

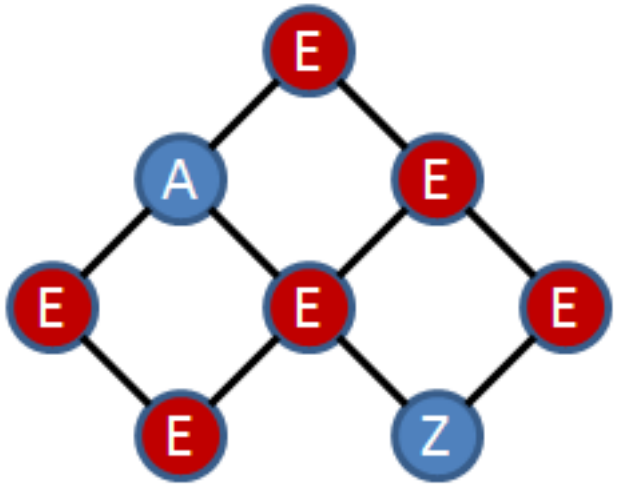
The three Markov properties are not equivalent to each other at all.

In fact, the Local Markov property is stronger than the Pairwise one, while weaker than the Global one.

Pairwise Markov Property

Pairwise Markov property: Any two non-adjacent variables are conditionally independent given all other variables:

$$X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}} \text{ if } \{u,v\} \notin E$$

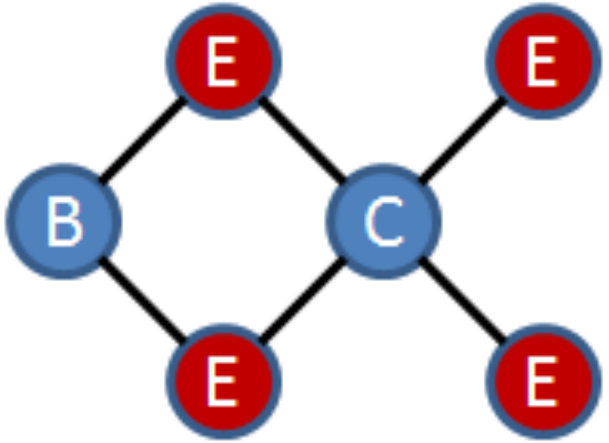


Local Markov Property

Local Markov property: A variable is conditionally independent of all other variables given its neighbors:

$$X_v \perp\!\!\!\perp X_{V \setminus cl(v)} \mid X_{ne(v)}$$

where ne(v) is the set of neighbors of v, and cl(v) = {v} ∪ ne(v) is the closed neighbourhood of v.

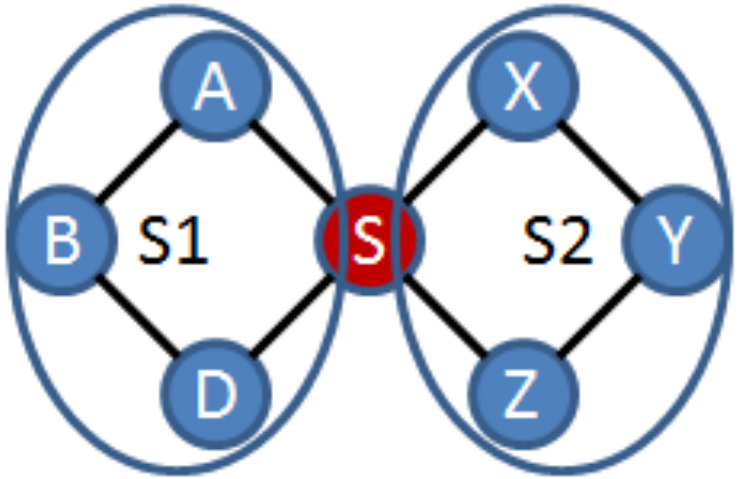


Global Markov property

Global Markov property: Any two subsets of variables are conditionally independent given a separating subset:

$$X_A \perp\!\!\!\perp X_B \mid X_S$$

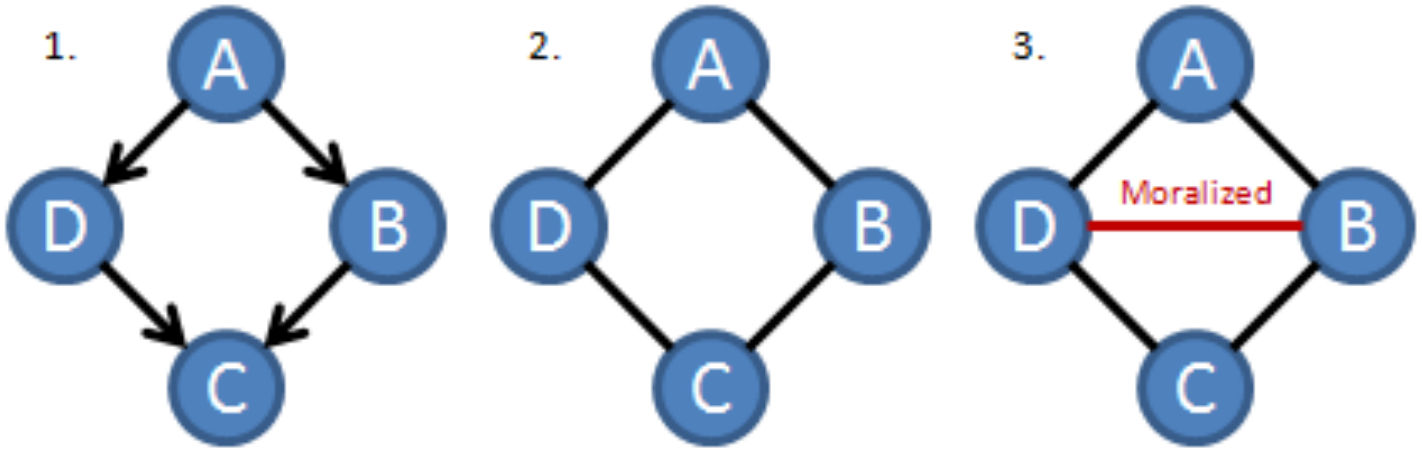
where every path from a node in A to a node in B passes through S.



Bayesian to Markov

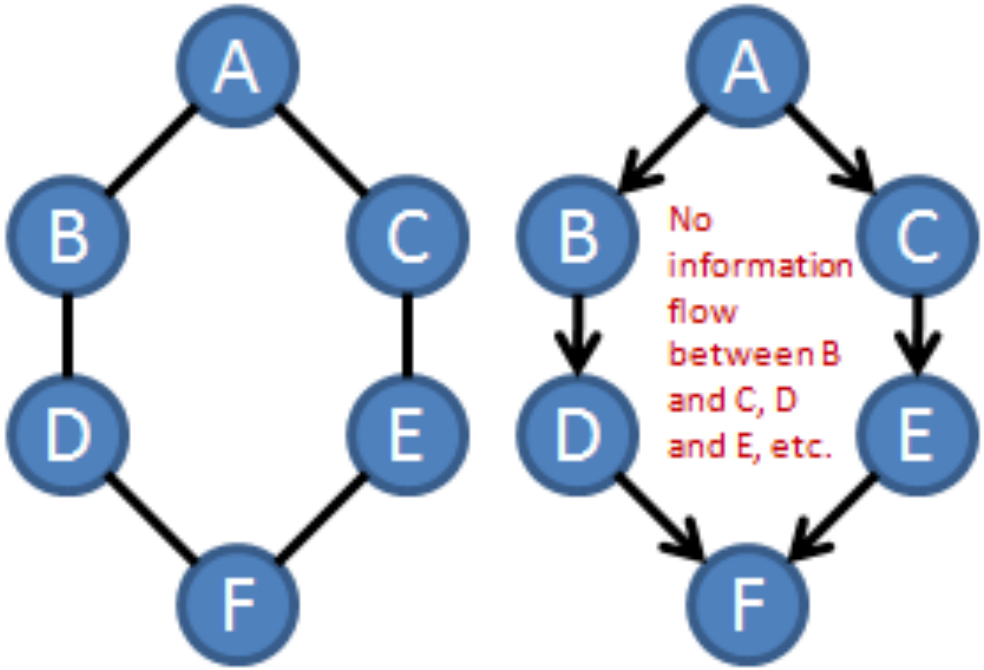
To convert from a Bayesian network to a Markov network we follow three steps:

1. Maintain the structure of the Bayesian network.
2. Eliminate directionality by making all edges undirected.
3. Moralize



Markov to Bayesian

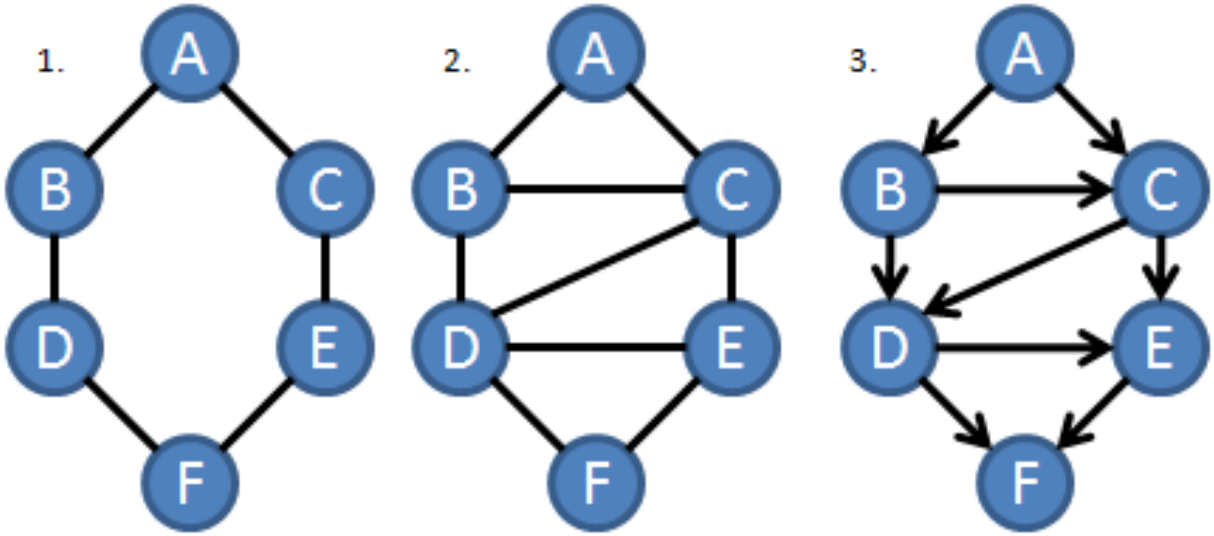
- Data flow problems



Markov to Bayesian Solution

To overcomes this, you perform three steps:

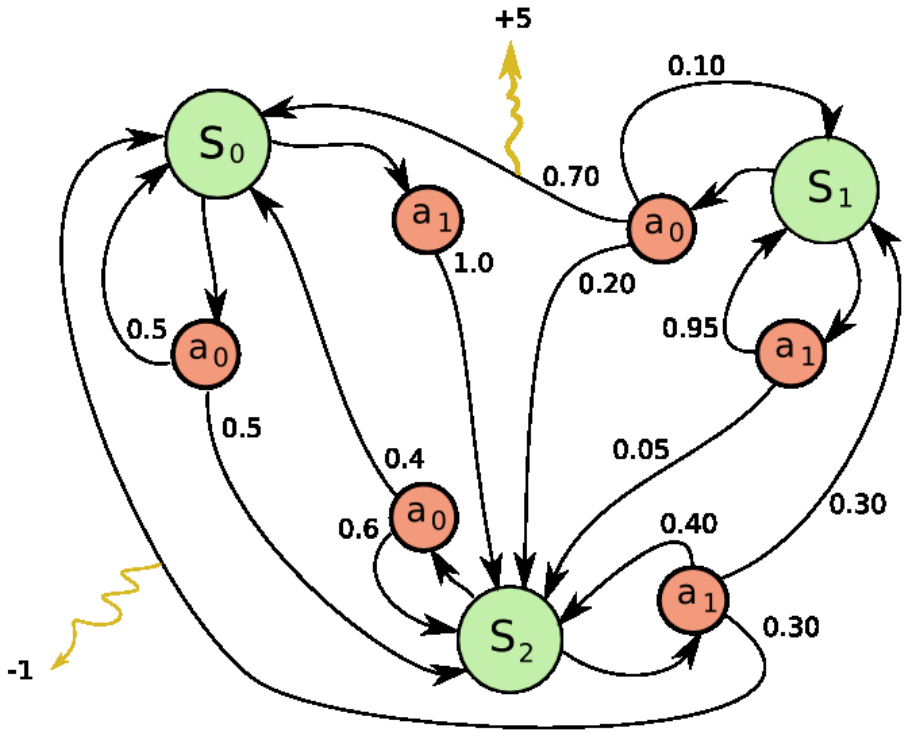
1. Maintain the structure of the Markov network.
2. Triangulate the graph to guarantee all dependency representations.
3. Add directionality.



Start again

Markov Decision Processes

http://en.wikipedia.org/wiki/Markov_decision_process



"Markov Decision Process example" by MistWiz –
Own work. Licensed under Public domain via Wikimedia Commons –
http://commons.wikimedia.org/wiki/File:Markov_Decision_Process_example.png#mediaviewer/File:Markov_Decision_Process_example.png

MDP Definition

A Markov decision process is a 4-tuple $(S,A,P_{\cdot}(\cdot,\cdot),R_{\cdot}(\cdot,\cdot))$, where

- S is a finite set of states,
- A is a finite set of actions (alternatively, A_s is the finite set of actions available from state s),
- $P_a(s,s') = Pr(s_{t+1}=s' \mid s_t = s, a_t=a)$ is the probability that action a in state s at time t will lead to state s' at time $t+1$,
- $R_a(s,s')$ is the immediate reward (or expected immediate reward) received after transition to state s' from state s.

(Note: The theory of Markov decision processes does not state that S or A are finite, but the basic algorithms below assume that they are finite.)

Problem

The core problem of MDPs is to find a "policy" for the decision maker: a function π that specifies the action $\pi(s)$ that the decision maker will choose when in state s.

Note that once a Markov decision process is combined with a policy in this way, this fixes the action for each state and the resulting combination behaves like a Markov chain.

Policy

The goal is to choose a policy π that will maximize some cumulative function of the random rewards, typically the expected discounted sum over a potentially infinite horizon:

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \qquad \text{(where we choose } a_t = \pi(s_t))$$

- where γ is the discount factor and satisfies $0 \leq \gamma < 1$. (For example, $\gamma = 1/(1+r)$ when the discount rate is r.) γ is typically close to 1.

Because of the Markov property, the optimal policy for this particular problem can indeed be written as a function of *s* only, as assumed above.

Algorithms

MDPs can be solved by linear programming or dynamic programming.

We present the latter approach.

Suppose we know the state transition function P and the reward function R, and we wish to calculate the policy that maximizes the expected discounted reward.

The standard family of algorithms to calculate this optimal policy requires storage for two arrays indexed by state:

value V, which contains real values, and

policy π which contains actions.

At the end of the algorithm, π will contain the solution and $V(s)$ will contain the discounted sum of the rewards to be earned (on average) by following that solution from state s.

The Steps

The algorithm has the following two kinds of steps, which are repeated in some order for all the states until no further changes take place. They are defined recursively as follows:

$$\pi(s) := \operatorname*{argmax}_a \left\{ \sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V(s')) \right\}$$

$$V(s) := \sum_{s'} P_{\pi(s)}(s,s')(R_{\pi(s)}(s,s') + \gamma V(s'))$$

Their order depends on the variant of the algorithm; one can also do them for all states at once or state by state, and more often to some states than others. As long as no state is permanently excluded from either of the steps, the algorithm will eventually arrive at the correct solution.

Partial Observability

The solution above assumes that the state s is known when action is to be taken; otherwise $\pi(s)$ cannot be calculated. When this assumption is not true, the problem is called a partially observable Markov decision process or POMDP.

http://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process

Burnetas, A.N. and M. N. Katehakis. "Optimal Adaptive Policies for Markov Decision Processes, Mathematics of Operations Research, 22,(1), 1995.

Reinforcement Learning

If the probabilities or rewards are unknown, the problem is one of reinforcement learning (Sutton and Barto, 1998).

For this purpose it is useful to define a further function, which corresponds to taking the action a and then continuing optimally (or according to whatever policy one currently has):

$$Q(s,a) = \sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V(s'))$$

While this function is also unknown, experience during learning is based on (s, a) pairs (together with the outcome s'); that is, "I was in state s and I tried doing a and s' happened"). Thus, one has an array Q and uses experience to update it directly. This is known as Q-learning.

Sutton, R. S. and Barto A. G. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA, 1998.

Reinforcement Learning

Reinforcement learning can solve Markov decision processes without explicit specification of the transition probabilities; the values of the transition probabilities are needed in value and policy iteration.

In reinforcement learning, instead of explicit specification of the transition probabilities, the transition probabilities are accessed through a simulator that is typically restarted many times from a uniformly random initial state.

Reinforcement learning can also be combined with function approximation to address problems with a very large number of states.

References

http://en.wikipedia.org/wiki/Markov_random_field

https://vv.cs.byu.edu/facwiki/scraped-cs677sp10/cs677sp10/index.php/Undirected_Models.html#Monte_Carlo_Markov_Chain_.28MCMC.29

http://www.cs.princeton.edu/courses/archive/spr06/cos598C/papers/AndrieuFreitasDoucetJordan2003.pdf

http://en.wikipedia.org/wiki/Markov_decision_process

Questions you should be able to answer – Day 1

- What is the intuition behind Bayes theorem?
- What is the difference between a maximum a posteriori hypothesis and a maximum likelihood hypothesis and a minimum description length hypothesis?
- What are the benefits and drawbacks of naïve bayes?
- What is a Markov chain is and what is the Markov Property?
- What is difference between a hidden Markov model and a Markov chain?

Questions you should be able to answer – Day 2

- What is a MCMC algorithm?
- What is Monte Carlo Tree Search?
- What is the difference between a Markov Random Field and a Bayesian Network?
- What is a Markov Decision Process?
- What is a Partially Observable Markov Decision Process?