

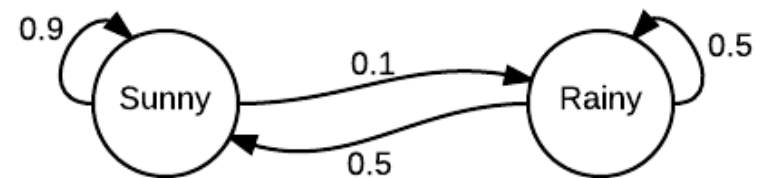
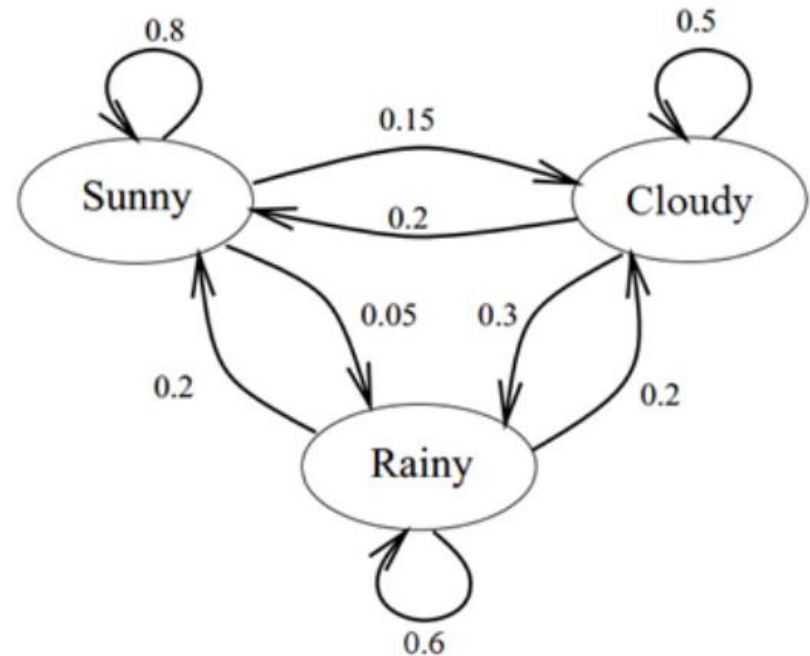
Introduction to Hidden Markov Models

Presented by: Derek Kane



Overview of Topics

- ❖ Introduction to Markov Models
- ❖ Hidden Markov Models
- ❖ Forward Algorithm
- ❖ Viterbi Algorithm
- ❖ Baum-Welch Algorithm
- ❖ Practical Example
 - ❖ Stock Market



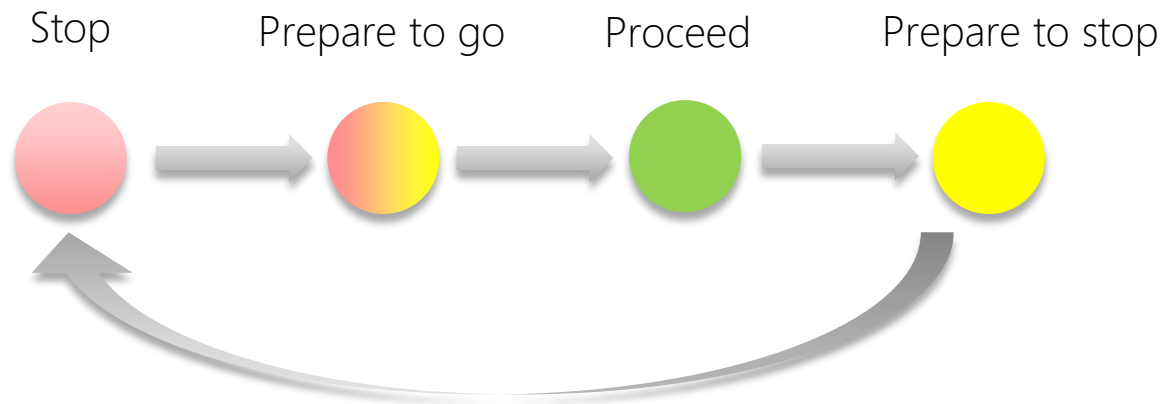
Introduction to Markov Processes



- ❖ Often we are interested in finding patterns which appear over a space of time.
- ❖ These patterns are observed in many areas in the nature.
 - ❖ The pattern of commands someone uses in instructing a computer.
 - ❖ Sequences of words in sentences.
 - ❖ The sequence of phonemes in spoken words.
 - ❖ Popcorn popping in an air popper.
- ❖ What we are most interested towards understanding with sequence of events processes is whether or not this sequence produces useful patterns which can be modeled.

Introduction to Markov Processes

- ❖ Let's first consider a set of traffic lights; the sequence of lights is red - red/amber - green - amber - red. The sequence can be pictured as a state machine (trellis diagram), where the different states of the traffic lights follow each other.



- ❖ Notice that each state is dependent solely on the previous state, so if the lights are green, an amber light will always follow - that is, the system is deterministic. Deterministic systems are relatively easy to understand and analyze, once the transitions are fully known.
- ❖ This system is referred to as an observable Markov process.

Introduction to Markov Processes

- ❖ Now let's consider the simple example of someone trying to deduce the weather from a piece of seaweed.
- ❖ Folklore tells us that "soggy" seaweed means wet weather, while "dry" seaweed means sun.
- ❖ If the seaweed is in an intermediate state ("damp"), then we cannot be sure.
- ❖ However, the state of the weather is not restricted to the state of the seaweed, so we may say on the basis of an examination that the weather is probably raining or sunny.



Introduction to Markov Processes



- ❖ A second useful clue would be the state of the weather on the preceding day (or, at least, its probable state).
- ❖ By combining knowledge about what happened yesterday with the observed seaweed state, we might come to a better forecast for today.
- ❖ This is typical of the type of system we will consider in this tutorial.

Introduction to Markov Processes

- ❖ First we will introduce systems which generate probabilistic patterns in time, such as the weather fluctuating between sunny and rainy.
- ❖ We then look at systems where what we wish to predict is not what we observe - the underlying system is hidden.
- ❖ In our previous example, the observed sequence would be the seaweed and the hidden system would be the actual weather.



Introduction to Markov Processes



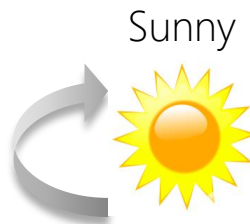
- ❖ We then will look at some problems that can be solved once the system has been modeled. For the above example, we may want to know:
- ❖ What the weather was for a week given each day's seaweed observation.
- ❖ Given a sequence of seaweed observations, is it winter or summer? Intuitively, if the seaweed has been dry for a while it may be summer, if it has been soggy for a while it might be winter.

Introduction to Markov Processes

- ❖ How would our weather example look like as a Markov process? Lets build some trellis diagrams based off of the observable weather.
- ❖ Imagine we have 3 states of weather: Sunny, Cloudy, and Rainy. Each state depends only upon the previous state.
- ❖ Here is a possible sequence with each sequence taking a single day.

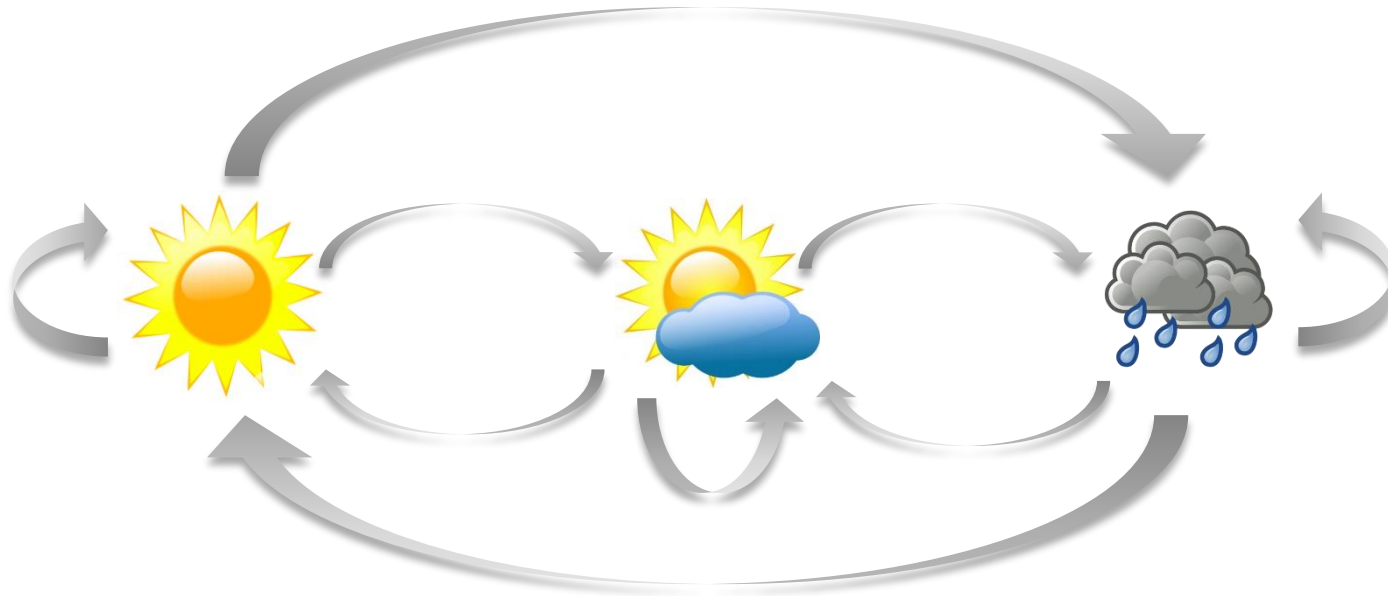


- ❖ This is another possible sequence. One sunny day could then lead to another sunny day. This loop is often referred to as a self transition.



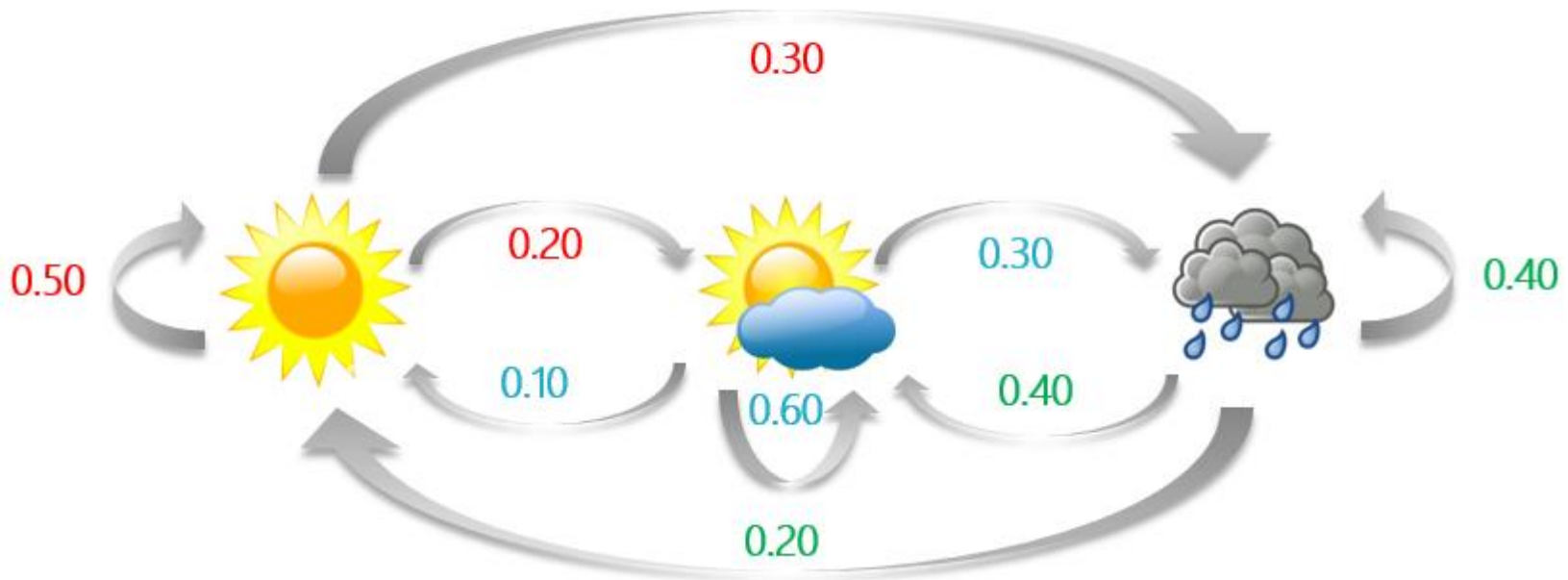
Introduction to Markov Processes

- ❖ We can continue to build all of the observable possibilities until we reach the following diagram.



Introduction to Markov Processes

- ❖ We can also construct the probabilities for these various weather patterns based upon observations. Ex. The probability of the next day being sunny after a sunny day is **0.50**.

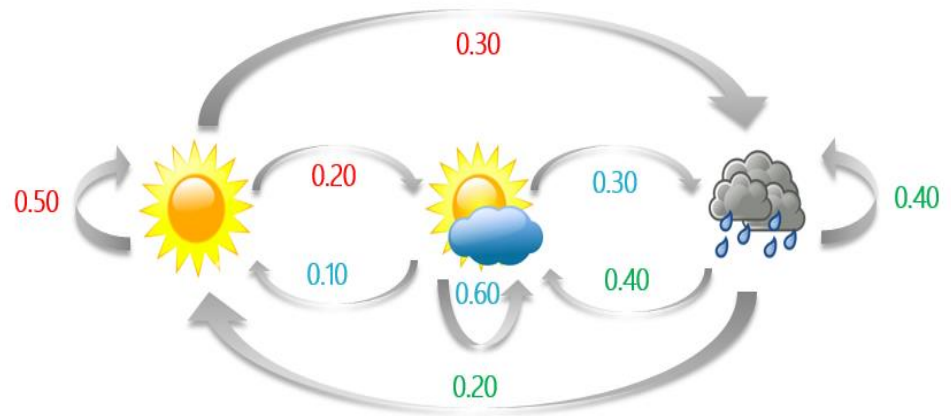


- ❖ **Note:** The total sum of each color probabilities is equal to 1.

Introduction to Markov Processes

- ❖ This system of observed probabilities can be represented in a matrix format. This matrix is also referred to as a transition matrix.

		Weather Today		
		Sunny	Cloudy	Rainy
Weather Yesterday	Sunny	0.50	0.20	0.30
	Cloudy	0.10	0.60	0.30
	Rainy	0.20	0.40	0.40



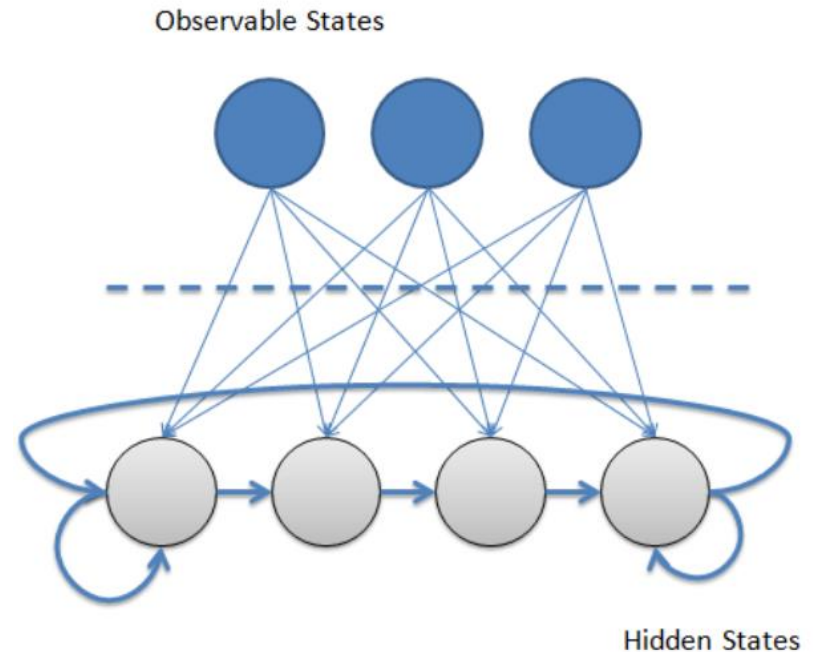
Introduction to Markov Processes



- ❖ In some cases the patterns that we wish to find are not described sufficiently by a Markov process.
- ❖ Returning to the weather example, a hermit may perhaps not have access to direct weather observations, but does have a piece of seaweed.
- ❖ Folklore tells us that the state of the seaweed is probabilistically related to the state of the weather - the weather and seaweed states are closely linked.
- ❖ In this case we have two sets of states, the observable states (the state of the seaweed) and the hidden states (the state of the weather).
- ❖ We wish to devise an algorithm for the hermit to forecast weather from the seaweed and the Markov assumption without actually ever seeing the weather.

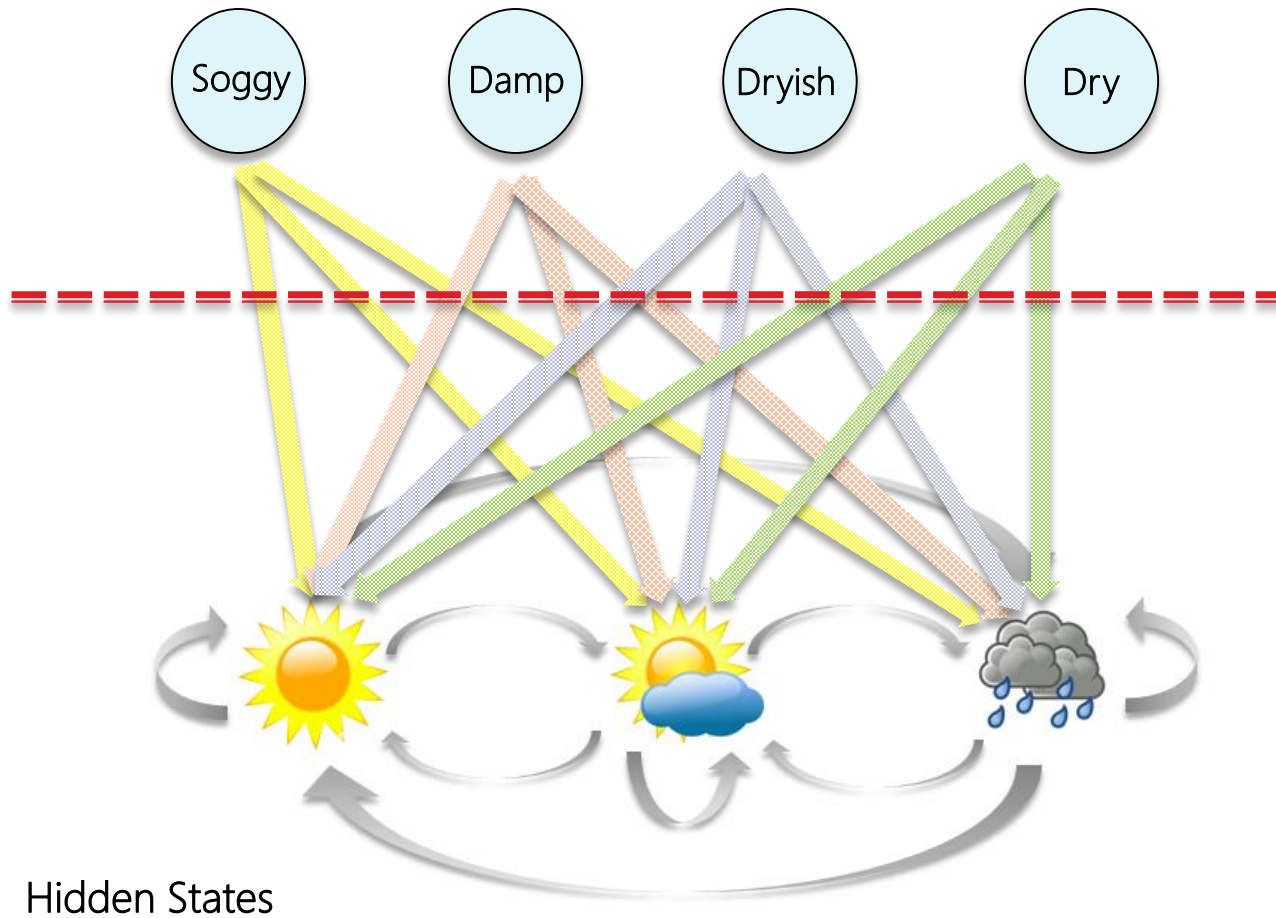
Hidden Markov Models

- ❖ In such cases the observed sequence of states is probabilistically related to the hidden process.
- ❖ We model such processes using a hidden Markov model where there is an underlying hidden Markov process changing over time, and a set of observable states which are related somehow to the hidden states.
- ❖ It is important to note that the number of states in the hidden process and the number of observable states may be different.
- ❖ In a three state weather system (sunny, cloudy, rainy) it may be possible to observe four grades of seaweed dampness (dry, dryish, damp, soggy).



Introduction to Markov Processes

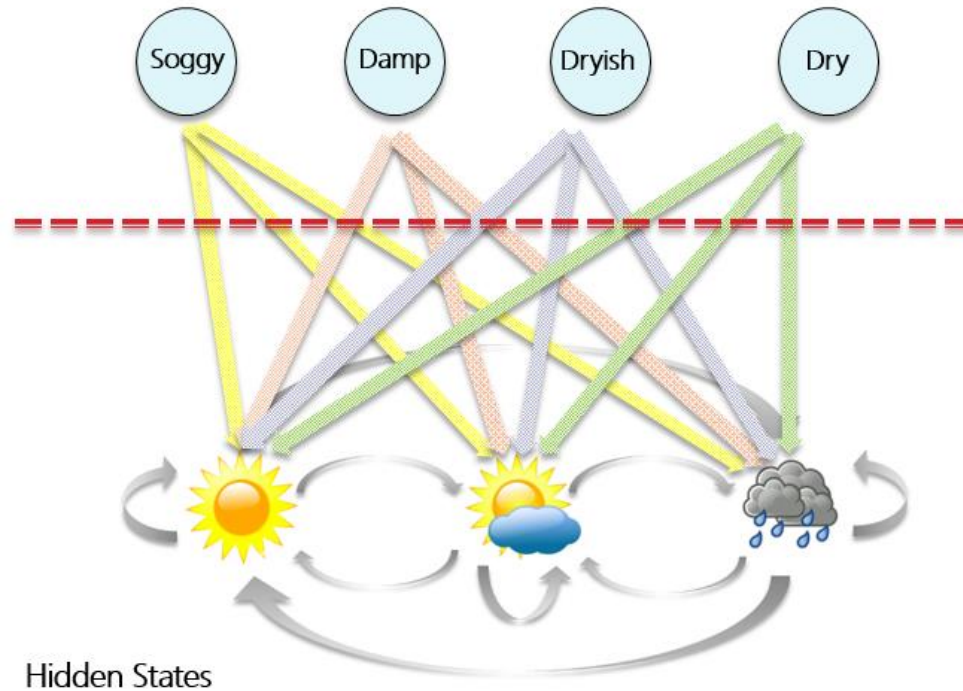
Observable States



Hidden States

Hidden Markov Models

Observable States



- ❖ The diagram shows the hidden and observable states in the weather example.
- ❖ It is assumed that the hidden states (the true weather) are modelled by a simple first order Markov process, and so they are all connected to each other.
- ❖ The connections between the hidden states and the observable states represent the probability of generating a particular observed state given that the Markov process is in a particular hidden state.
- ❖ It should be clear that all probabilities "entering" an observable state will sum to 1, since in the above case it would be the sum of $\Pr(\text{Obs}|\text{Sun})$, $\Pr(\text{Obs}|\text{Cloud})$ and $\Pr(\text{Obs}|\text{Rain})$.

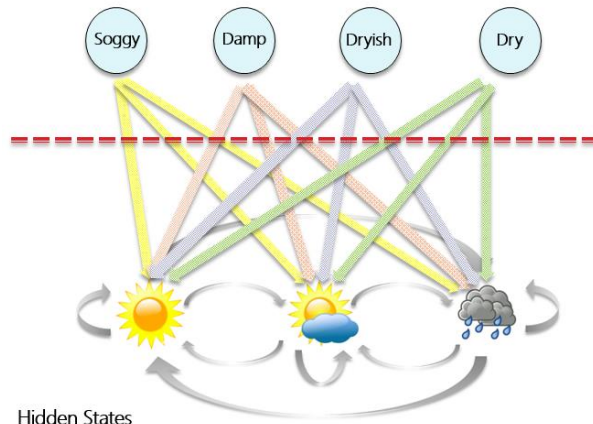
Hidden Markov Models

Seaweed State Today

Weather
Yesterday

	Dry	Dryish	Damp	Soggy
Sunny	0.60	0.20	0.15	0.05
Cloudy	0.25	0.25	0.25	0.25
Rainy	0.05	0.10	0.35	0.50

Observable States

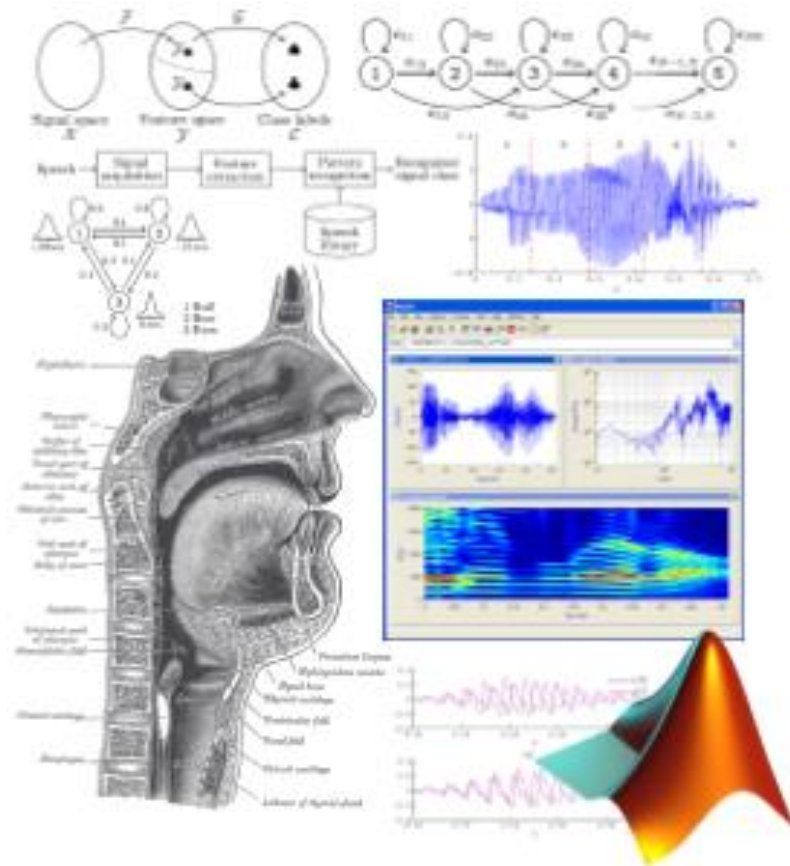


- ❖ In addition to the probabilities defining the Markov process, we therefore have another matrix, termed the confusion matrix or emission matrix, which contains the probabilities of the observable states given a particular hidden state.
- ❖ For the weather example the emission matrix is shown to the left. Notice that the sum of each matrix row is 1.
- ❖ Each probability in the state transition matrix and in the emission matrix is time independent - that is, the matrices do not change structure in time as the system evolves.
- ❖ In practice, this is one of the most *unrealistic assumptions* of Markov models about real processes.

Hidden Markov Models

Once a system can be described as a HMM, three problems can be solved.

- ❖ Evaluation
 - ❖ Decoding
 - ❖ Learning
- ❖ The first two are pattern recognition problems:
- ❖ Finding the probability of an observed sequence given a HMM (evaluation)
 - ❖ Finding the sequence of hidden states that most probably generated an observed sequence (decoding).
- ❖ The third problem is generating a HMM given a sequence of observations (learning).



Hidden Markov Models

Evaluation

- ❖ Consider the problem where we have a number of HMMs describing different systems, and a sequence of observations.
- ❖ We may want to know which HMM most probably generated the given sequence.
- ❖ For example, we may have a “Summer” model and a “Winter” model for the seaweed, since behavior is likely to be different from season to season - we may then hope to determine the season on the basis of a sequence of dampness observations.
- ❖ We use the forward algorithm to calculate the probability of an observation sequence given a particular HMM, and hence choose the most probable HMM.

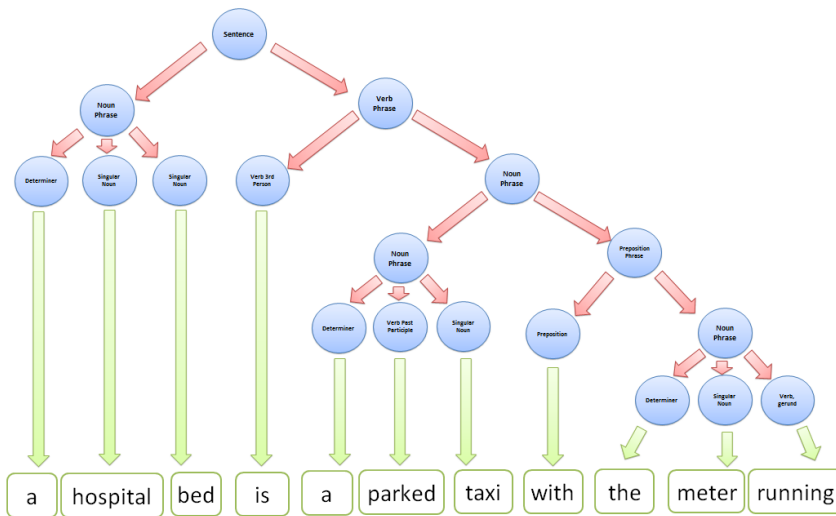


Hidden Markov Models



Decoding

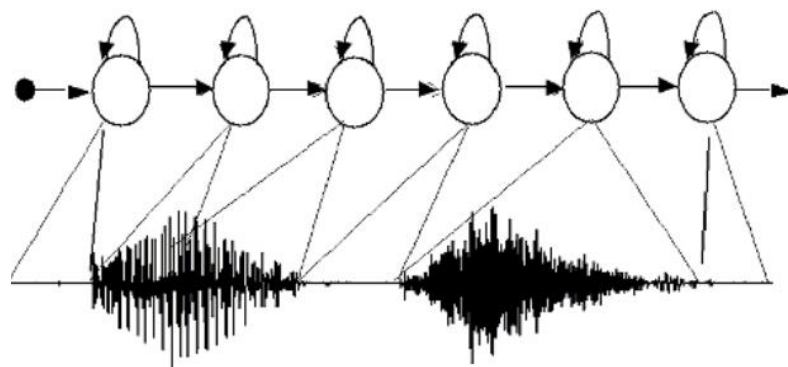
- ❖ Finding the most probable sequence of hidden states given some observations.
- ❖ The question usually of most interest with HMM is to find the hidden states that generated the observed output.
- ❖ In many cases we are interested in the hidden states of the model since they represent something of value that is not directly observable.
- ❖ Consider the example of the seaweed and the weather; a blind hermit can only sense the seaweed state, but needs to know the weather.
- ❖ We use the Viterbi algorithm to determine the most probable sequence of hidden states given a sequence of observations and a HMM.



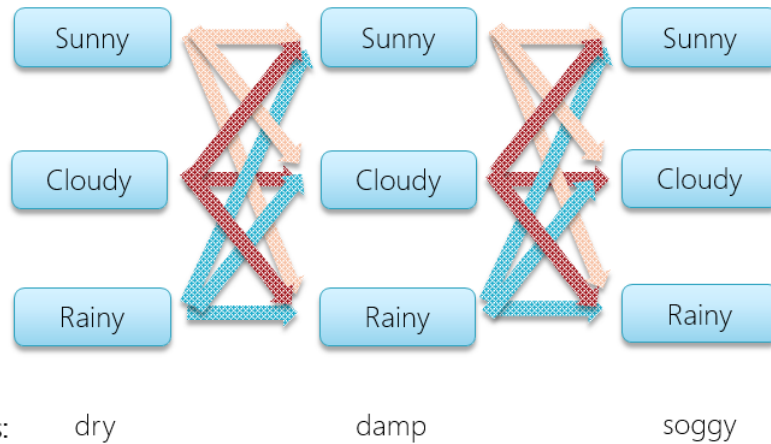
Hidden Markov Models

Learning

- ❖ Involves generating a HMM from a sequence of observations.
- ❖ The hardest problem associated with HMMs is to take a sequence of observations (from a known set), known to represent a set of hidden states, and fit the most probable HMM.
- ❖ We can use the Baum-Welch expectation maximization (EM) algorithm to identify local optimal parameters for the HMM.



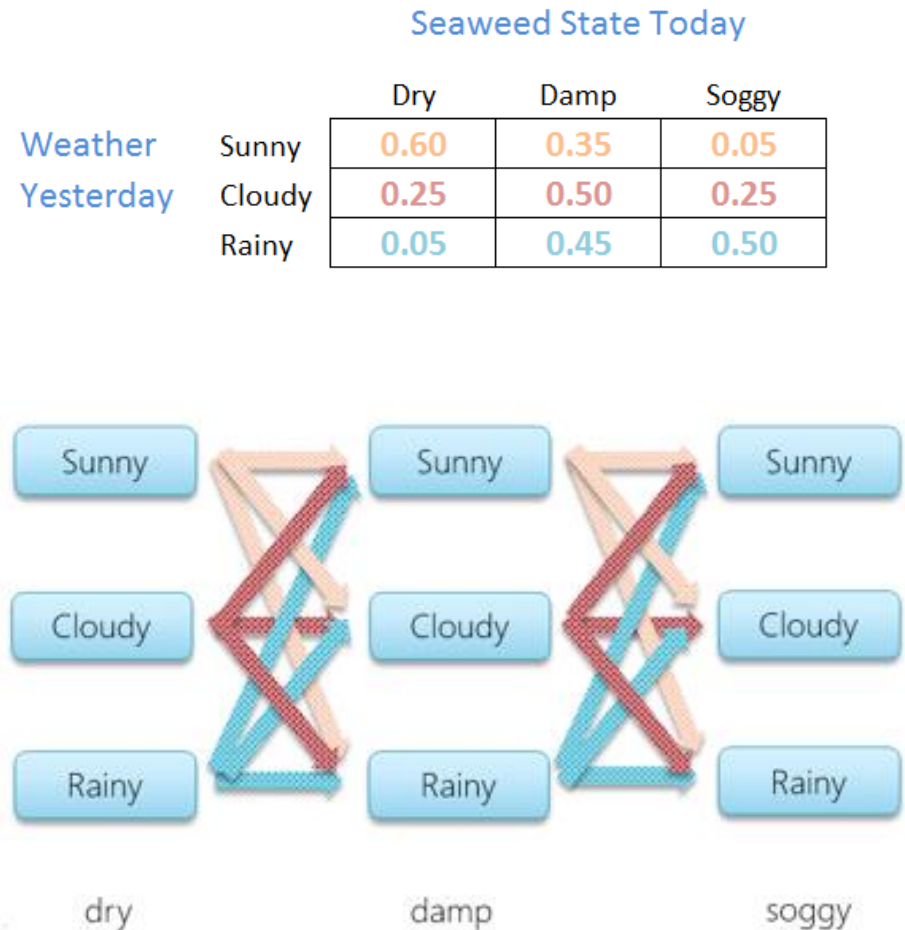
Forward Algorithm – Exhaustive Search



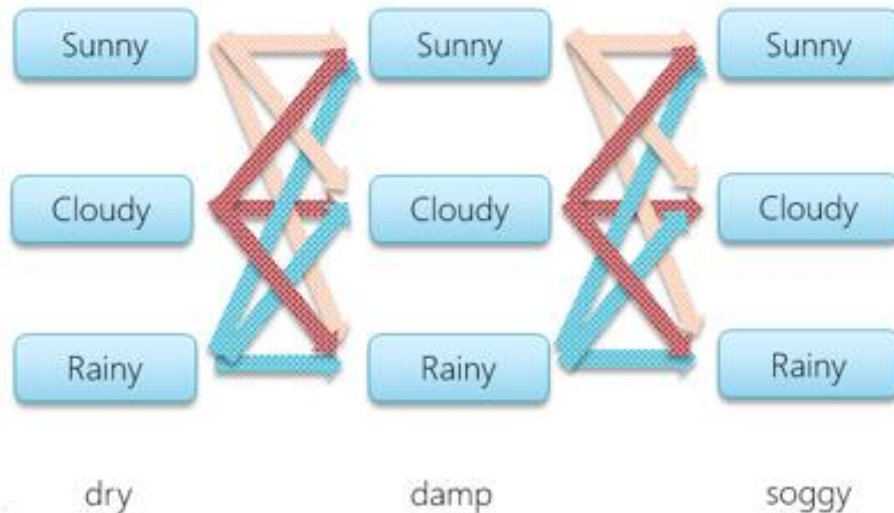
- ❖ There are a number of ways to find the probability of an observed sequence.
- ❖ We want to find the probability of an observed sequence given an HMM, where the parameters are known.
- ❖ Consider the weather example; we have a HMM describing the weather and its relation to the state of the seaweed, 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.

Forward Algorithm – Exhaustive Search

- ❖ 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 emission matrix.



Forward Algorithm – Exhaustive Search

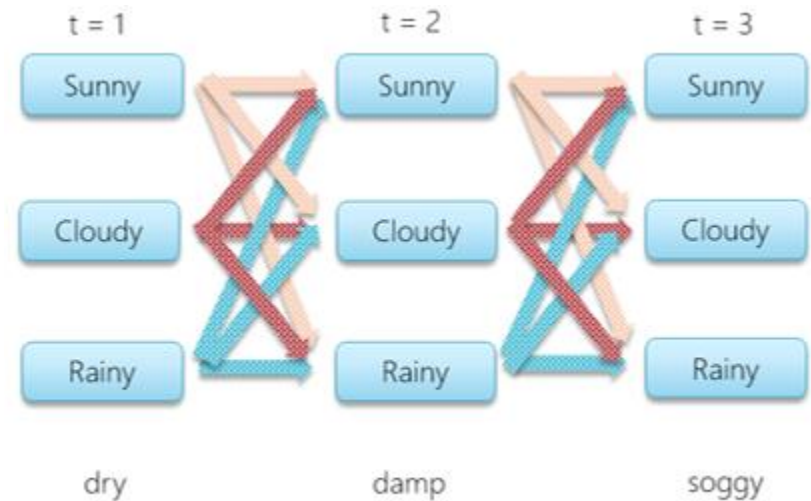


- ❖ It can be seen that one method of calculating the probability of the observed sequence would be to find each possible sequence of the hidden states, and sum these probabilities.
- ❖ For the example, there would be $3^3=27$ possible different weather sequences, and so the probability is:
- ❖
$$\Pr(\text{dry,damp,soggy} \mid \text{HMM}) = \Pr(\text{dry,damp,soggy} \mid \text{sunny,sunny,sunny}) + \Pr(\text{dry,damp,soggy} \mid \text{sunny,sunny,cloudy}) + \dots \Pr(\text{dry,damp,soggy} \mid \text{rainy,rainy,rainy})$$
- ❖ Calculating the probability in this manner is computationally expensive, particularly with large models, and we find that we can use the time invariance of the probabilities to reduce the complexity of the problem.

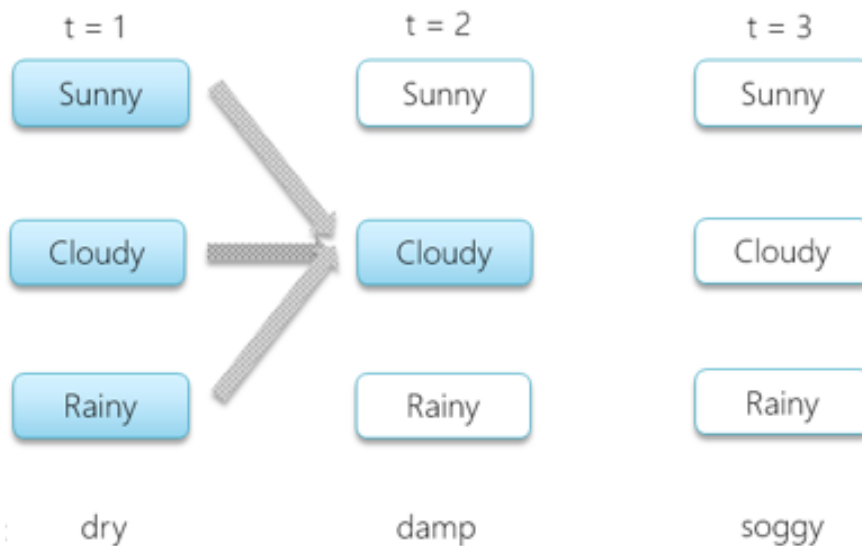
Forward Algorithm – Complexity Reduction

- ❖ Now let's take a look at the 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:

$$Y^{(k)} = (Y_{k_1}, Y_{k_2}, \dots, Y_{k_T})$$



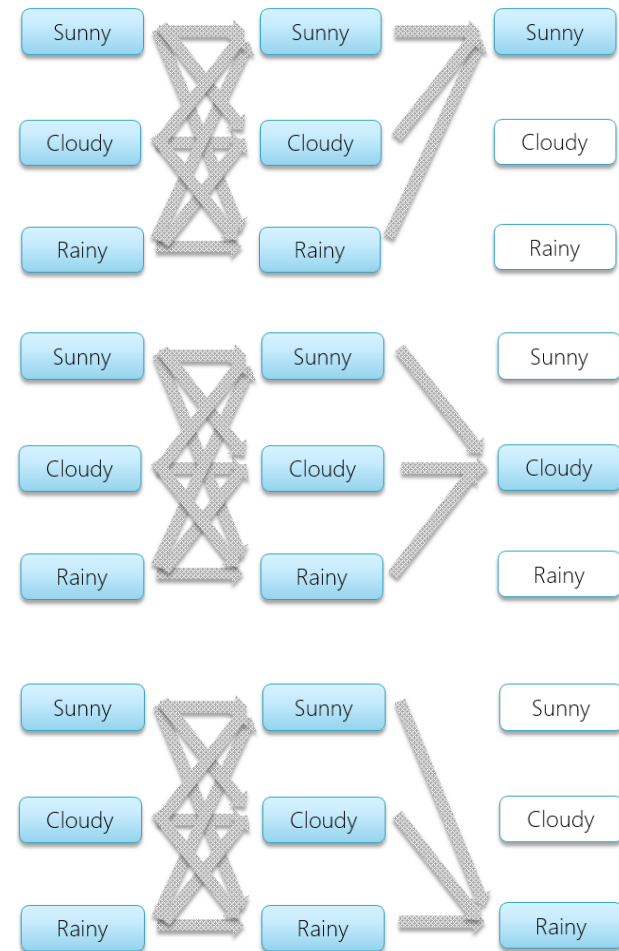
Forward Algorithm – Complexity Reduction



- ❖ Consider the trellis showing the states and first order transitions for the observation sequence dry, damp, soggy:
- ❖ 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.

Forward Algorithm – Complexity Reduction

- ❖ We denote the partial probability of state j at time t as $\alpha_t(j)$ - this partial probability is calculated as:
- ❖ $\alpha_t(j) = \Pr(\text{observation} \mid \text{hidden state is } j) * \Pr(\text{all paths to state } j \text{ at time } t)$
- ❖ The partial probabilities for the final observation hold the probability of reaching those states going through all possible paths.
- ❖ The final partial probabilities are calculated from the following paths of our seaweed trellis diagram.
- ❖ 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.



Forward Algorithm – Complexity Reduction

We calculate partial probabilities as:

- ❖ $\alpha_t(j) = \Pr(\text{observation} \mid \text{hidden state is } j) \times \Pr(\text{all paths to state } j \text{ at time } t)$
- ❖ In the special case where $t = 1$, there are no paths to the state. The probability of being in a state at $t = 1$ is therefore the initial probability, i.e. $\Pr(\text{state} \mid t = 1) = P(\text{state})$, and we therefore calculate partial probabilities at $t = 1$ as this probability multiplied by the associated observation probability:

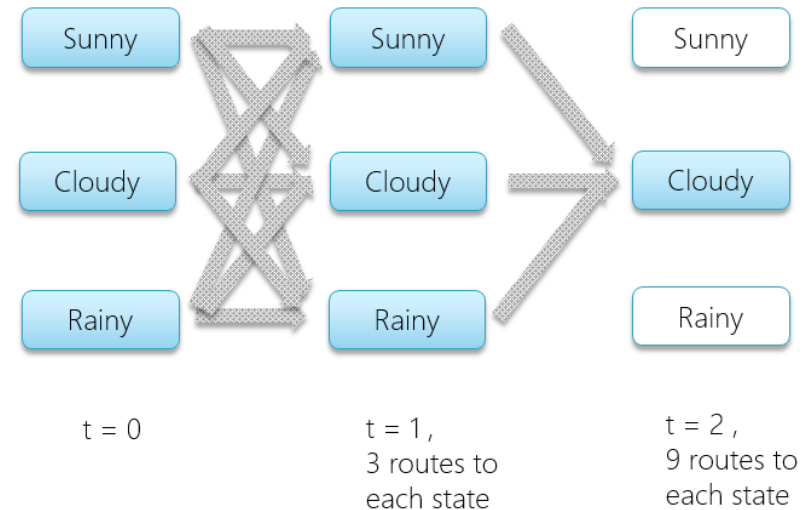
$$\alpha_1(j) = \pi(j) * B_{jk_1}$$

- ❖ Thus the probability of being in state j at initialization is dependent on that state's probability together with the probability of observing what we see at that time.

Forward Algorithm – Complexity Reduction

Calculating α 's at time, t (> 1):

- ❖ We recall that a partial probability is calculated as : $\alpha_t(j) = \Pr(\text{observation} \mid \text{hidden state is } j) \times \Pr(\text{all paths to state } j \text{ at time } t)$.
- ❖ We can assume (recursively) that the first term of the product is available, and now consider the term $\Pr(\text{all paths to state } j \text{ at time } t)$.
- ❖ To calculate the probability of getting to a state through all paths, we can calculate the probability of each path to that state and sum them.



Forward Algorithm – Complexity Reduction

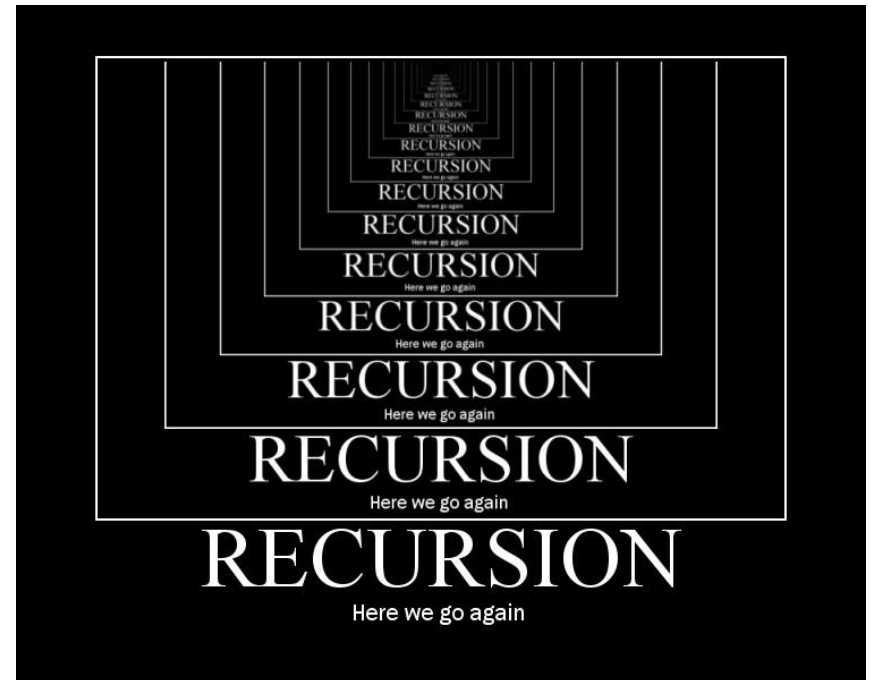
- ❖ The number of paths needed to calculate α increases exponentially as the length of the observation sequence increases but the α 's at time $t-1$ give the probability of reaching that state through all previous paths.
- ❖ We can define α 's at time t in terms of those at time $t-1$ -i.e.,

$$\alpha_{t+1}(j) = b_{jk_{t+1}} \sum_{i=1}^n \alpha_t(i) \alpha_{ij}$$

- ❖ Therefore we calculate the probabilities as the product of the appropriate observation probability (that is, that state j provoked what is actually seen at time $t+1$) with the sum of probabilities of reaching that state at that time - this latter comes from the transition probabilities together with α from the preceding stage.

Forward Algorithm – Complexity Reduction

- ❖ Notice that we have an expression to calculate α at time $t+1$ using only the partial probabilities at time t .
- ❖ We can now calculate the probability of an observation sequence given a HMM recursively - i.e. we use α 's at $t=1$ to calculate α 's at $t=2$; α 's at $t=2$ to calculate α 's at $t=3$; and so on until $t = T$.
- ❖ The probability of the sequence given the HMM is then the sum of the partial probabilities at time $t = T$



Forward Algorithm – Complexity Reduction

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 with n hidden states.
- ❖ An exhaustive evaluation would involve computing for all possible execution sequences.

$$X_i = (X_{i_1}, X_{i_2}, \dots \dots \dots, X_{i_T})$$

- ❖ the quantity:

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

- ❖ which sums the probability of observing what we do - note that the load 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 load will only be linear in T .

Forward Algorithm

Summary:

- ❖ Our aim is to find the probability of a sequence of observations given a HMM - ($\Pr(\text{observations} \mid I)$).
- ❖ We reduce the complexity of calculating this probability by first calculating partial probabilities (α 's). These represent the probability of getting to a particular state, s , at time t .
- ❖ We then see that at time $t = 1$, the partial probabilities are calculated using the initial probabilities (from the P vector) and $\Pr(\text{observation} \mid \text{state})$ (from the confusion matrix); also, the partial probabilities at time $t (> 1)$ 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 $t = 1, 2, \dots, T$, and adding all a '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.

Forward Algorithm

Forward algorithm definition:

- ❖ We use the forward algorithm to calculate the probability of a T long observation sequence:

$$Y^{(k)} = (Y_{k_1}, Y_{k_2}, \dots \dots \dots, Y_{k_T})$$

- ❖ where each of the y is one of the observable set. Intermediate probabilities (a's) are calculated recursively by first calculating a for all states at t=1.

$$\alpha_1(j) = \pi(j) * B_{jk_1}$$

Forward Algorithm

- ❖ Then for each time step, $t = 2, \dots, T$, the partial probability α is calculated for each state:

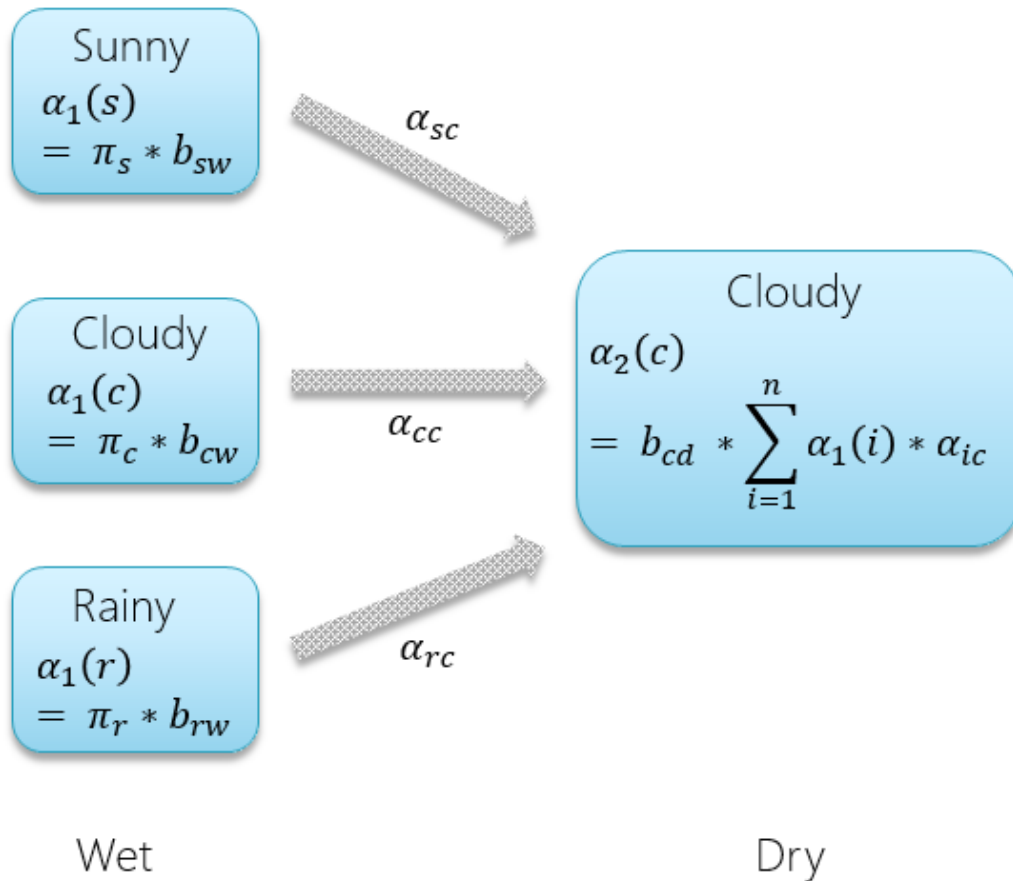
$$\alpha_{t+1}(j) = b_{jk_t} \sum_{i=1}^n \alpha_t(i) \alpha_{ij}$$

- ❖ The product of the appropriate observation probability and the sum over all possible routes to that state, exploiting recursion by knowing these values already for the previous time step.
- ❖ Finally the sum of all partial probabilities gives the probability of the observation, given the HMM.

$$\Pr(Y^{(k)}) = \sum_{j=1}^n \alpha_T(j)$$

- ❖ To recap, each partial probability (at time $t > 2$) is calculated from all the previous states.

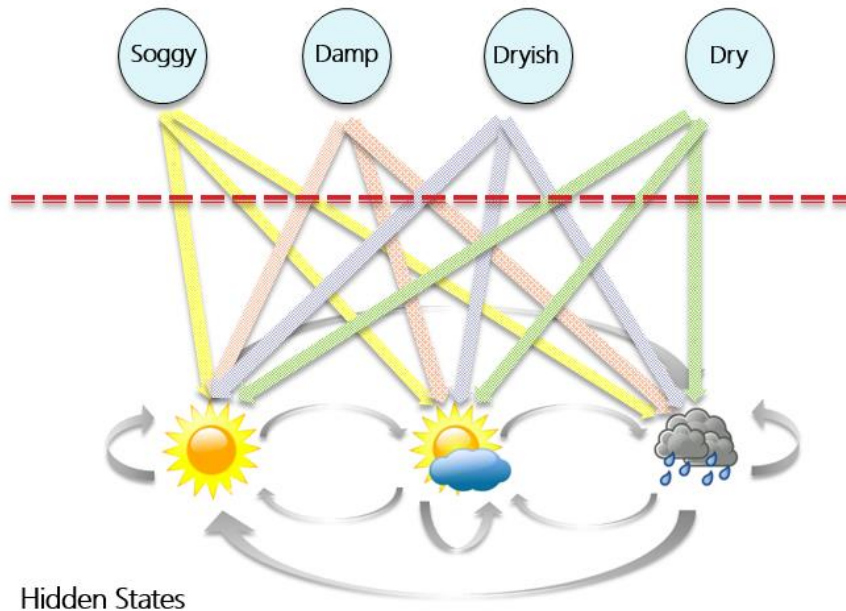
Forward Algorithm



- ❖ Using the 'weather' example, the diagram shows the calculation for α at $t = 2$ for the cloudy state.
- ❖ This is the product of the appropriate observation probability b and the sum of the previous partial probabilities multiplied by the transition probabilities α .

Forward Algorithm

Observable States



Hidden States

Initial State
Probabilities

Sunny	0.63
Cloudy	0.17
Rainy	0.20

Forward Algorithm

Confusion Matrix

		Observed States			
		Dry	Dryish	Damp	Soggy
Hidden States	Sunny	0.60	0.20	0.15	0.05
	Cloudy	0.25	0.25	0.25	0.25
	Rainy	0.05	0.10	0.35	0.50

State Transition Matrix

		Weather Today		
		Sunny	Cloudy	Rainy
Weather Yesterday	Sunny	0.500	0.375	0.125
	Cloudy	0.250	0.125	0.625
	Rainy	0.250	0.375	0.375

Summary

- ❖ We use the forward algorithm to find the probability of an observed sequence given a HMM. It exploits recursion in the calculations to avoid the necessity for exhaustive calculation of all paths through the execution trellis.
- ❖ Given this algorithm, it is straightforward to determine which of a number of HMMs best describes a given observation sequence - the forward algorithm is evaluated for each, and that giving the highest probability selected.

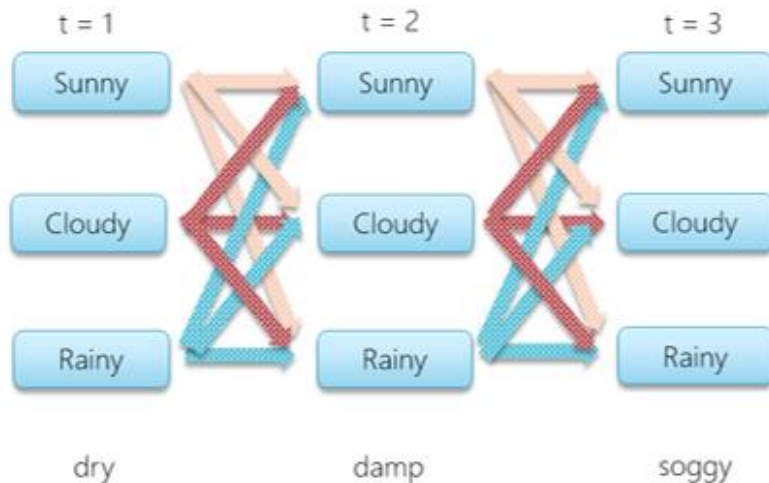
Viterbi Algorithm

- ❖ Now that we have a solid understanding of the Forward algorithm used in evaluation techniques, let's shift focus and review a decoding HMM scenario with the Viterbi Algorithm.
- ❖ 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.
- ❖ The Viterbi algorithm is designed to help solve this form of problem.



Andrew Viterbi

Viterbi Algorithm – Exhaustive Search

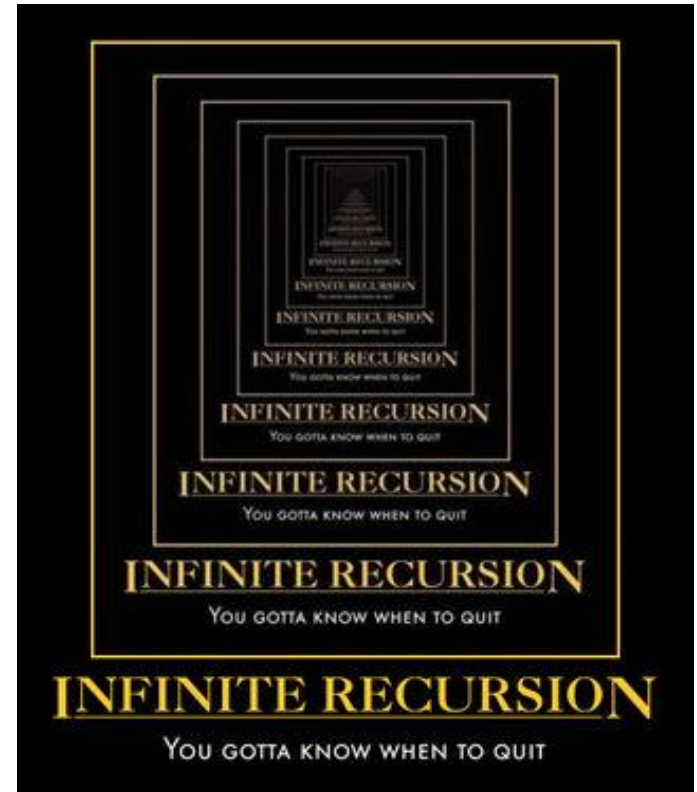


- ❖ We can use a picture of the execution trellis to visualize the relationship between states and observations.
- ❖ We can find the most probable sequence of hidden states by listing all possible sequences of hidden states and finding the probability of the observed sequence for each of the combinations.
- ❖ The most probable sequence of hidden states is that combination that maximizes $\Pr(\text{observed sequence} \mid \text{hidden state combination})$.
- ❖ As with the forward algorithm, we can use the time invariance of the probabilities to reduce the complexity of the calculation.

Viterbi Algorithm – Reducing Complexity

Reducing complexity using recursion

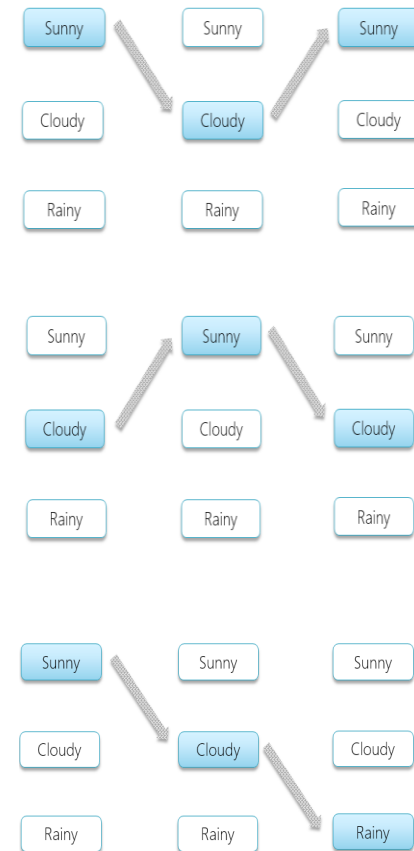
- ❖ We will consider recursively finding the most probable sequence of hidden states given an observation sequence and a HMM.
- ❖ We will first define the partial probability δ , which is the probability of reaching a particular intermediate state in the trellis.
- ❖ We then show how these partial probabilities 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.



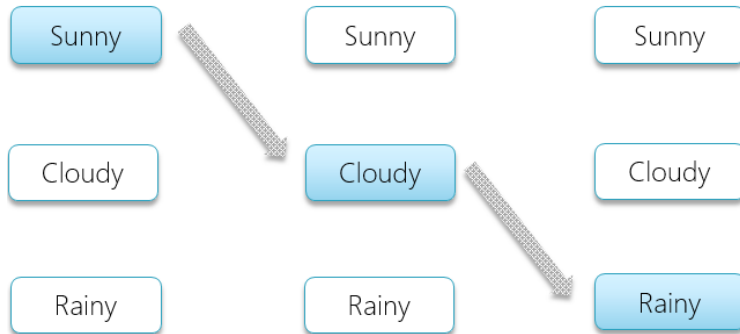
Viterbi Algorithm – Reducing Complexity

Partial probabilities (δ 's) and partial best paths

- ❖ Consider the trellis from a couple slides ago showing the states and first order transitions for the observation sequence dry, damp, or soggy.
- ❖ 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.



Viterbi Algorithm – Reducing Complexity



- ❖ 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 probability 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.

Viterbi Algorithm – Reducing Complexity

- ❖ 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.



Viterbi Algorithm – Reducing Complexity

Calculating δ 's at time $t = 1$

- ❖ We calculate the δ partial probabilities as the most probable route to our current position (given particular knowledge such as observation and probabilities of the previous state).
- ❖ When $t = 1$ the most probable path to a state does not sensibly exist; however we use the probability of being in that state given $t = 1$ and the observable state k_1 .

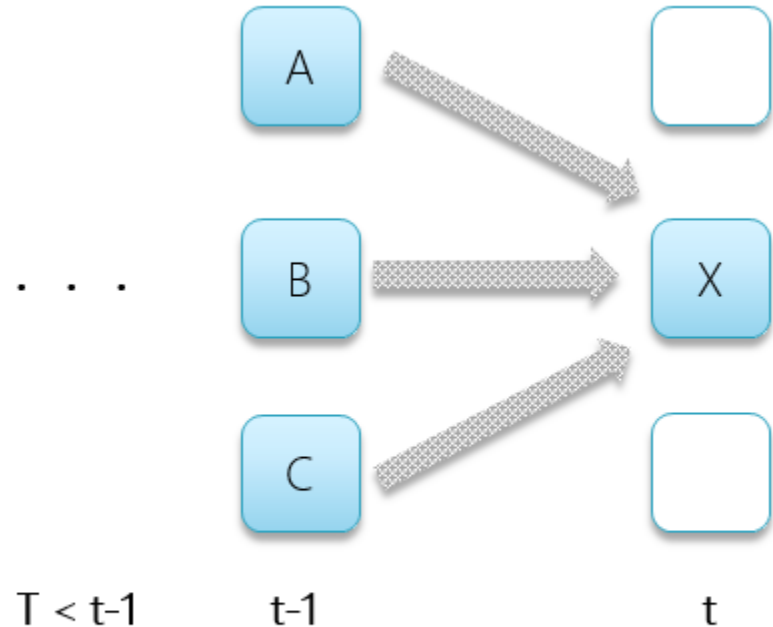
$$\delta_1(i) = \pi(i)b_{bk_1}$$

- ❖ As in the forward algorithm, this quantity is compounded by the appropriate observation probability.

Viterbi Algorithm – Reducing Complexity

Calculating δ 's at time t (>1)

- ❖ We now show that the partial probabilities δ at time t can be calculated in terms of the δ 's at time $t-1$.
- ❖ Lets consider the trellis to the right.
- ❖ We consider calculating the most probable path to the state X at time t ; this path to X will have to pass through one of the states A , B or C at time $(t-1)$.



Viterbi Algorithm – Reducing Complexity

- ❖ Therefore the most probable path to X will be one of:
 - ❖ (sequence of states), . . . , A, X
 - ❖ (sequence of states), . . . , B, X
 - ❖ (sequence of states), . . . , C, X
- ❖ We want to find the path ending AX, BX or CX which has the maximum probability.
- ❖ Recall that the Markov assumption says that the probability of a state occurring given a previous state sequence depends only on the previous n states.
- ❖ In particular, with a first order Markov assumption, the probability of X occurring after a sequence depends only on the previous state, i.e.
- ❖ $\Pr(\text{most probable path to A}) = \Pr(X | A) * \Pr(\text{observation} | X)$

Viterbi Algorithm – Reducing Complexity

- ❖ Following this, the most probable path ending AX will be the most probable path to A followed by X. Similarly, the probability of this path will be: $\Pr(\text{most probable path to A}) = \Pr(X | A) * \Pr(\text{observation} | X)$.
- ❖ So, the probability of the most probable path to X is :

$$\Pr(X \text{ at time } t) = \max_{i=A_i B_i C_i} \Pr(i \text{ at time } (t - 1)) * \Pr(X|i) * \Pr(\text{obs. at time } t | X)$$

- ❖ where the first term is given by δ at t-1, the second by the transition probabilities and the third by the observation probabilities.

Viterbi Algorithm – Reducing Complexity

- ❖ Here is a generalization of the previous formula with the probability of the partial best path to a state i at time t when the observation k_t is seen, is:

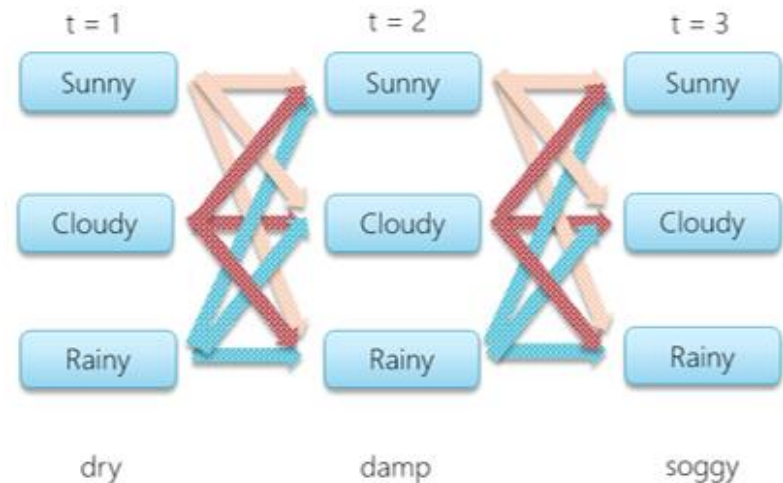
$$\delta_t(i) = \max_j (\delta_{t-1}(j) a_{ij} b_{i k_t})$$

- ❖ Here, we are assuming knowledge of the previous state, using the transition probabilities and multiplying by the appropriate observation probability.
- ❖ We then select the maximum probability.

Viterbi Algorithm – Reducing Complexity

Back pointers, ϕ 's

- ❖ Consider the trellis to the right.
- ❖ At each intermediate and end state we know the partial probability, $\delta(i,t)$.
- ❖ However the aim is to find the most probable sequence of states through the trellis given an observation sequence - therefore we need some way of remembering the partial best paths through the trellis.



Viterbi Algorithm – Reducing Complexity

- ❖ Recall that to calculate the partial probability, δ at time t we only need the δ 's for time $t-1$.
- ❖ Having calculated this partial probability, it is thus possible to record which preceding state was the one to generate $\delta(i,t)$ - that is, in what state the system must have been at time $t-1$ if it is to arrive optimally at state i at time t .
- ❖ This recording (remembering) is done by holding for each state a back pointer ϕ which points to the predecessor that optimally provokes the current state.
- ❖ Formally, we can write:

$$\phi_t(i) = \operatorname{argmax}_j (\delta_{t-1}(j) a_{ij})$$

- ❖ Here, the argmax operator selects the index j which maximizes the bracketed expression.
- 

Viterbi Algorithm – Reducing Complexity

- ❖ Lets take a look at this formula again:

$$\phi_t(i) = \operatorname{argmax}_j (\delta_{t-1}(j) a_{ij})$$

- ❖ Notice that this expression is calculated from the δ 's of the preceding time step and the transition probabilities, and does not include the observation probability (unlike the calculation of the δ 's themselves).
- ❖ This is because we want these ϕ 's to answer the question:
 - ❖ "If I am here, by what route is it most likely I arrived?"
 - ❖ This question relates to the hidden states, and therefore confusing factors due to the observations can be overlooked.

Viterbi Algorithm

Advantages of the approach

- ❖ Using the Viterbi algorithm to decode an observation sequence carries two important advantages:
- ❖ There is a reduction in computational complexity by using the recursion - this argument is exactly analogous to that used in justifying the forward algorithm.
- ❖ The Viterbi algorithm has the very useful property of providing the **best interpretation** given the entire context of the observations. An alternative to it might be, for example, to decide on the execution sequence.



Viterbi Algorithm

Lets consider the following:

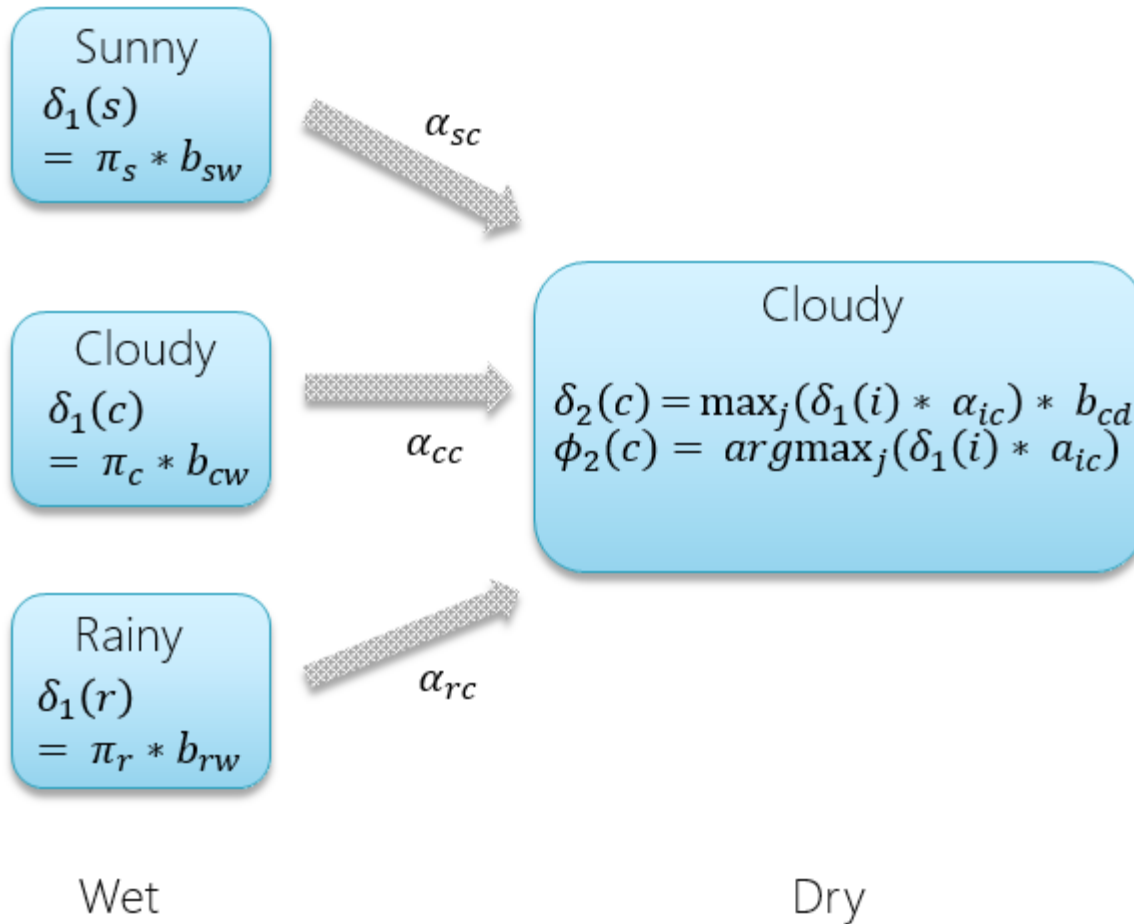
$$X_i = (X_{i_1}, X_{i_2}, \dots \dots \dots, X_{i_T})$$

❖ Where

$$i_1 = \operatorname{argmax}_j (\pi(j) b_{jk_1})$$
$$i_t = \operatorname{argmax}_j (a_{i_{t-1}k_t} b_{jk_t})$$

- ❖ In this case, decisions are taken about a likely interpretation in a 'left-to-right' manner, with an interpretation being guessed given an interpretation of the preceding stage.
- ❖ This approach, in the event of a noise garble half way through the sequence, will wander away from the correct answer.
- ❖ Conversely, the Viterbi algorithm will look at the whole sequence before deciding on the most likely final state, and then "backtracking" through the ϕ 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.

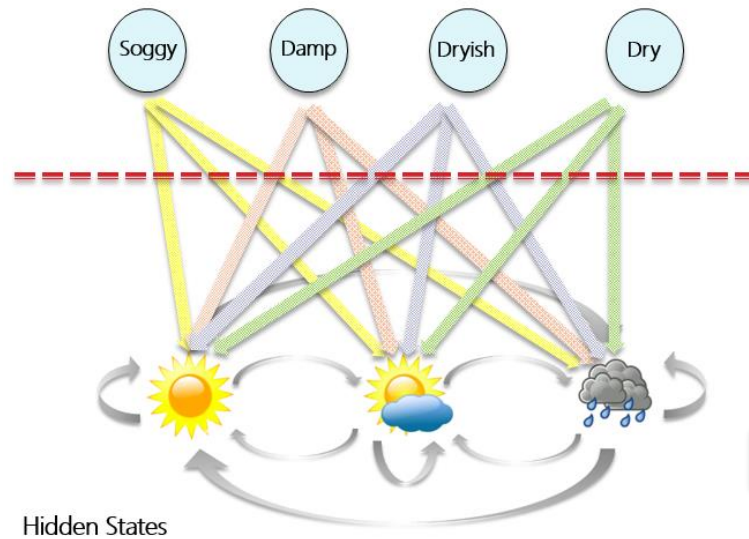
Viterbi Algorithm



- ❖ Using the “weather” example, the diagram shows the calculation for individual δ and ϕ and for the cloudy state.
- ❖ The calculation of δ 's is similar to the calculation of partial probability (α 's) in the forward algorithm.

Viterbi Algorithm

Observable States



Initial State
Probabilities

Sunny	0.63
Cloudy	0.17
Rainy	0.20

Confusion Matrix

Observed States

		Observed States			
		Dry	Dryish	Damp	Soggy
Hidden States	Sunny	0.60	0.20	0.15	0.05
	Cloudy	0.25	0.25	0.25	0.25
	Rainy	0.05	0.10	0.35	0.50

State Transition Matrix

Weather Today

Weather
Yesterday

		Sunny	Cloudy	Rainy
		0.500	0.250	0.250
Weather Yesterday	Sunny	0.500	0.250	0.250
	Cloudy	0.375	0.125	0.375
	Rainy	0.125	0.675	0.375

Baum-Welch EM Algorithm

- ❖ 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 P 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.
- ❖ This is what the Baum-Welch Algorithm is designed to resolve.



Lloyd R. Welch

Baum-Welch EM Algorithm

- ❖ Imagine that we have a dataset of observations and we want the parameters of a Hidden Markov Model that generated that data.
- ❖ How do we leverage a Hidden Markov model for prediction when we do not have an understanding of the relationships between the observed and hidden states?
- ❖ The problem is that there is no known way to analytically solve for the model which maximizes the probability of the observation sequence. The calculation space is too large for an exact calculation.
- ❖ This issue can be circumvented through the use of heuristic methods. While a heuristic methodology only creates a partial solution, the results are extremely useful. The result of this work is the creation of locally optimal parameters.

Note: The Baum-Welch EM algorithm is more complex in nature than the forward algorithm and the Viterbi algorithm. For this reason, it will not be presented here in full.



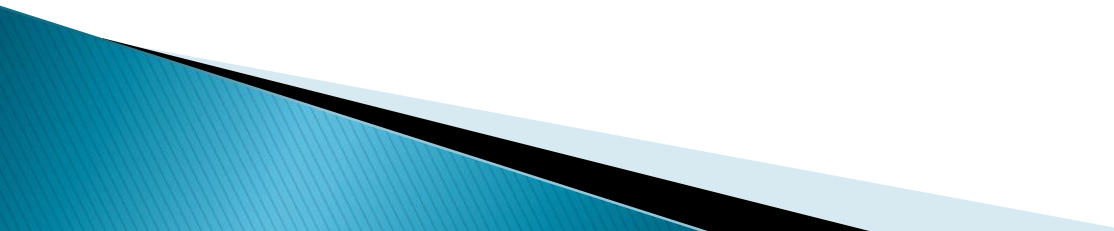
Baum-Welch EM Algorithm



- ❖ The transmission, emission, and initial distribution probabilities are all unknown.
- ❖ We first must first set the parameters/probabilities to some initial values.
- ❖ These values can take be from the following:
 - ❖ Some form of a uniform distribution
 - ❖ Incorporate some prior knowledge
 - ❖ Purely Random values (cannot be flat)

Baum-Welch EM Algorithm

Figure out the probability of a hidden state transition.

- ❖ Postulate that transition at every single spot in every single observed sequence (separately).
 - ❖ Then we see how the probabilities compare to the best probabilities for those observed sequences.
 - ❖ Use this ratio to update the transition probability.
-
- ❖ We then follow a similar procedure for the emission symbol from the hidden state.
-
- ❖ The next step calculates the probabilities of the learning set and then we rinse and repeat.
-
- ❖ Successive iterations increase the $P(\text{data})$ and we stop the procedure when the probability stops increasing significantly and converges (usually measured as log-likelihood ratios.)
- 

Baum-Welch Example

- ❖ Here is a more practical way to think about the Baum Welch algorithm. Lets go back to our seaweed weatherman example.
- ❖ For discussion purposes, we are going to simplify the scenario to only wet or dry seaweed for our observed state.
- ❖ Our seaweed weatherman while walking on the shores finds either wet or dry seaweed.
- ❖ I also know that either sunny or rainy weather can be predicted from the seaweed. (This implies that there are only 2 hidden states; sunny or rainy.)



Baum-Welch Example

- ❖ With the two hidden states, I need to invent a transition matrix and an emission matrix:

		Weather Today				Seaweed Today	
		Sunny	Rainy			Wet	Dry
Weather Yesterday	Sunny	0.50	0.50	Weather Yesterday	Sunny	0.20	0.80
	Rainy	0.40	0.60		Rainy	0.90	0.10

- ❖ We also need to decide on the initial weather forecast probabilities. After careful consideration, I decided to go with $p(\text{Sunny}) = 0.3$ and $p(\text{Rainy}) = 0.7$.

Baum-Welch Example

Lets imagine the following set of 9 observations:

WW, WW, WW, WW, WD, DD, DW, WW, WW

W = Wet

D = Dry

Note: If a Wet/Wet observation came from Sunny->Rainy sequence, we would calculate the probability as: $0.3 * 0.2 * 0.5 * 0.9 = 0.027$

		Weather Today	
		Sunny	Rainy
Weather Yesterday	Sunny	0.50	0.50
	Rainy	0.40	0.60

		Seaweed Today	
		Wet	Dry
Weather Yesterday	Sunny	0.20	0.80
	Rainy	0.90	0.10

$p(\text{Sunny}) = 0.3$ and $p(\text{Rainy}) = 0.7$

Baum-Welch Example

- ❖ We will estimate our transition matrix:

Sequence	P(Seq) if Sunny-> Rainy	Best P(Seq)
WW	0.027	0.3403 Rainy -> Rainy
WW	0.027	0.3403 Rainy -> Rainy
WW	0.027	0.3403 Rainy -> Rainy
WW	0.027	0.3403 Rainy -> Rainy
WD	0.003	0.2016 Rainy -> Sunny
DD	0.012	0.096 Sunny -> Sunny
DW	0.108	0.108 Sunny -> Rainy
WW	0.027	0.3403 Rainy -> Rainy
WW	0.027	0.3403 Rainy -> Rainy
Total	0.285	2.4474

- ❖ Our estimate for the Sunny -> Rainy transition probability is now $0.285 / 2.4474 = 0.116$. Calculate the Rainy -> Sunny, Rainy -> Rainy, Sunny -> Sunny as well and normalize so that they all add up to 1 as needed, to update the transition matrix.

Baum-Welch Example

- ❖ Now we will estimate our new emission matrix:

Sequence	Best P(Seq) if Dry came from Sunny	Best P(Seq)
WD	0.2016 Rainy -> Sunny	0.2016 Rainy -> Sunny
DD	0.096 Sunny -> Sunny	0.096 Sunny -> Sunny
DW	0.108 Sunny -> Rainy	0.108 Sunny -> Rainy

The initial probabilities can also be updated through the following approach:

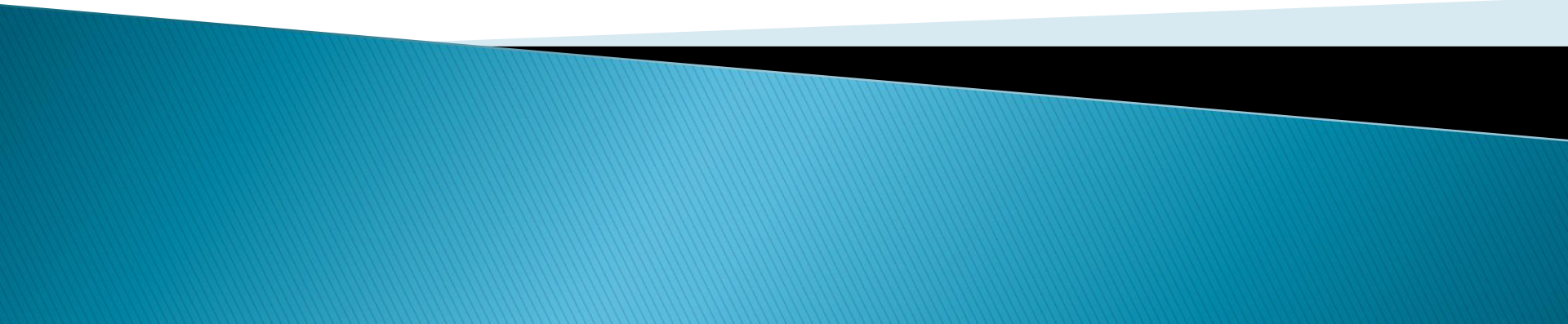
- ❖ Assume all sequences start with the hidden state "Sunny" and calculate the best probability.
- ❖ Assume all sequences start with the hidden state "Rainy" and calculate the best probability.
- ❖ Normalize to 1.

Baum-Welch Example

- ❖ This method has now created an updated transition, emission, and initial probabilities.
- ❖ We will repeat this method until the probabilities converge at which point the algorithm completes the run.
- ❖ **Note:** If we had guessed the wrong number of hidden states, it will be clear, though it is a bad strategy to through a huge range of possible hidden states to find the best model.
- ❖ This will lead to over-optimization.



Hidden Markov Model Predicting the Stock Market



Understanding the Data

- ❖ There are many applications for utilizing hidden markov models. The stock market is an example where we can draw from our knowledge of HMM's to model how the system behaves.
- ❖ The dataset will be scrapped off of the WWW using the quantmod library in R and contains information related a particular stock's performance.
- ❖ The goal for this exercise will be to decode the HMM using the Viterbi algorithm and then utilize the Baum-Welch EM algorithm to make a prediction for the next days value of the stock price.



Understanding the Data

- ❖ We will first extract information related to the stock TWII and then use the plot function in quantmod to review the stock data.



- ❖ The dataset contains information related to the opening and close stock price for a given day, the high and low price, and stock volume.

Understanding the Data

- ❖ First, let's take a look at the raw data in the table.

row.names	TWII.Open	TWII.High	TWII.Low	TWII.Close	TWII.Volume	TWII.Adjusted
1/1/03	6041.86	6078.17	5976.4	6033.47	2955000	6033.47
1/2/03	6081.71	6116.05	6075.35	6116.05	2964000	6116.05
1/3/03	6131.95	6131.95	6094	6098.78	2384200	6098.78
1/4/03	6141.28	6145.3	6123.84	6140.14	2365800	6140.14
1/5/03	6168.51	6168.51	6130.56	6141.14	2522800	6141.14
1/6/03	6157.47	6170	6140.34	6149.88	2575200	6149.88
1/7/03	6161.8	6167.9	6108.4	6119.06	2400000	6119.06
1/8/03	6155.38	6166.11	6130.99	6164.98	2486200	6164.98
1/9/03	6170.37	6186.12	6156.12	6169.08	2617600	6169.08
1/10/03	6159.97	6172.89	6132.99	6148.7	2165600	6148.7
1/11/03	6132.45	6139.99	6079.57	6082.56	2663600	6082.56
1/12/03	6080.19	6086.88	6015.56	6031.24	2717800	6031.24
1/13/03	6044.16	6050.4	6023.08	6035.59	2058000	6035.59
1/14/03	6015.98	6115.54	5994.46	6105.35	3146200	6105.35
1/15/03	6099.13	6121.62	6066.37	6067.34	2645800	6067.34
1/16/03	6017.25	6039.85	5966.16	5972.06	2942000	5972.06

- ❖ What we are interested in modeling with our HMM is the difference between the close value and the opening value for the current day.

Hidden Markov Model

- ❖ The first step will be to pull in our financial data from yahoo.

```
library(quantmod)
getSymbols("^TWII", src = "yahoo", from = "1900-01-01", to = "2015-12-31")

chartSeries(TWII, theme="black")

TWII_Subset <- window(TWII, start=as.Date("2013-01-01"), end = as.Date("2015-12-31"))
TWII_Train <- cbind(TWII_Subset$TWII.Close - TWII_Subset$TWII.Open)
```

- ❖ Notice that I sub-setted the data from 1/1/2013 to current
- ❖ Also, we will calculate the difference between the open and close values and store this dataset off for our model building.

Hidden Markov Model

- ❖ The next step will involve fitting our hidden markov model through the RHmm 2.0.3 package.
- ❖ For this example we are identifying 5 hidden states to the models with a series of different 4 distributions for each hidden state.
- ❖ Additionally, we are specifying a number of iterations for the algorithm to process to equal to 2000. This is to ensure convergence.

```
#####  
# Baum-Welch Algorithm  
#####  
  
library(RHmm)  
  
# Package Located at https://r-forge.r-project.org/R/?group\_id=85  
  
# hm_model <- HMMFit(obs=TWII_Train, nStates=5)  
  
hm_model <- HMMFit(obs=TWII_Train, dis="MIXTURE", nStates=5, nMixt=4,  
                  control=list(iter=2000))  
  
print(hm_model)
```


Hidden Markov Model

❖ Here is the output of the model:

Model:

5 states HMM with mixture of 4 gaussian distribution

Baum-Welch algorithm status:

Number of iterations : 533

Last relative variation of LLH function: 0.000001

Estimation:

Initial probabilities:

Pi 1	Pi 2	Pi 3	Pi 4	Pi 5
0	0	0	0	1

Transition matrix:

	State 1	State 2	State 3	State 4	State 5
State 1	2.475183e-01	2.037208e-16	9.323018e-02	6.592515e-01	1.210125e-38
State 2	4.419532e-01	8.240344e-07	4.841865e-73	2.215855e-01	3.364605e-01
State 3	1.344439e-46	9.462488e-29	5.729243e-01	2.071360e-31	4.270757e-01
State 4	1.499732e-02	8.532267e-01	1.060721e-01	2.570387e-02	2.111101e-11
State 5	1.898577e-22	3.652722e-19	2.904121e-34	1.000000e+00	1.851196e-23

Conditionnal distribution parameters:

Distribution parameters:

State 1

		mean	var	prop
mixt.	1	-63.92478	2.170185	0.03803497
mixt.	2	29.35526	37.419526	0.21161749
mixt.	3	-91.66134	78.445333	0.21868678
mixt.	4	-42.05702	70.152831	0.53166075

State 2

		mean	var	prop
mixt.	1	-77.41740	790.96972	0.1140821
mixt.	2	12.12229	28.54524	0.1538868
mixt.	3	52.04499	149.52386	0.2467862
mixt.	4	-13.29898	223.02129	0.4852449

Log-likelihood: -2721.07

BIC criterium: 5936.95

AIC criterium: 5600.14

1

Hidden Markov Model

- ❖ We will now utilize the Viterbi algorithm to determine the hidden states of the data.

```
#####  
# Viterbi Algorithm  
#####  
  
VitPath <- viterbi(hm_model, TWII_Train)  
  
#####  
# Predict the known values of the HMM model  
#####  
  
TWII_Predict <- cbind(TWII_Subset$TWII.Close, VitPath$states)
```

Lets graph the results

```
# Scatterplot  
  
chartSeries(TWII_Predict[,1])  
addTA(TWII_Predict[TWII_Predict[,2]==1,1],on=1,type="p",col=5,pch=25)  
addTA(TWII_Predict[TWII_Predict[,2]==2,1],on=1,type="p",col=6,pch=24)  
addTA(TWII_Predict[TWII_Predict[,2]==3,1],on=1,type="p",col=7,pch=23)  
addTA(TWII_Predict[TWII_Predict[,2]==4,1],on=1,type="p",col=8,pch=22)  
addTA(TWII_Predict[TWII_Predict[,2]==5,1],on=1,type="p",col=10,pch=21)
```

TWII_Predict dataframe

row.names	TWII.Close	Hidden State
1/1/03	6033.47	5
1/2/03	6116.05	4
1/3/03	6098.78	3
1/4/03	6140.14	1
1/5/03	6141.14	2
1/6/03	6149.88	5
1/7/03	6119.06	4
1/8/03	6164.98	4
1/9/03	6169.08	3
1/10/03	6148.7	1
1/11/03	6082.56	2
1/12/03	6031.24	2
1/13/03	6035.59	5
1/14/03	6105.35	4
1/15/03	6067.34	4
1/16/03	5972.06	4

Hidden Markov Model

- ❖ The scatterplot produces the following line chart in quantmod based upon the actual HWII stock data from 1/1/2013 to 2/17/2015.



Hidden Markov Model

- ❖ Now we will overlay the hidden states on top of the line chart to see the HMM represented graphically. Here is the first hidden state.



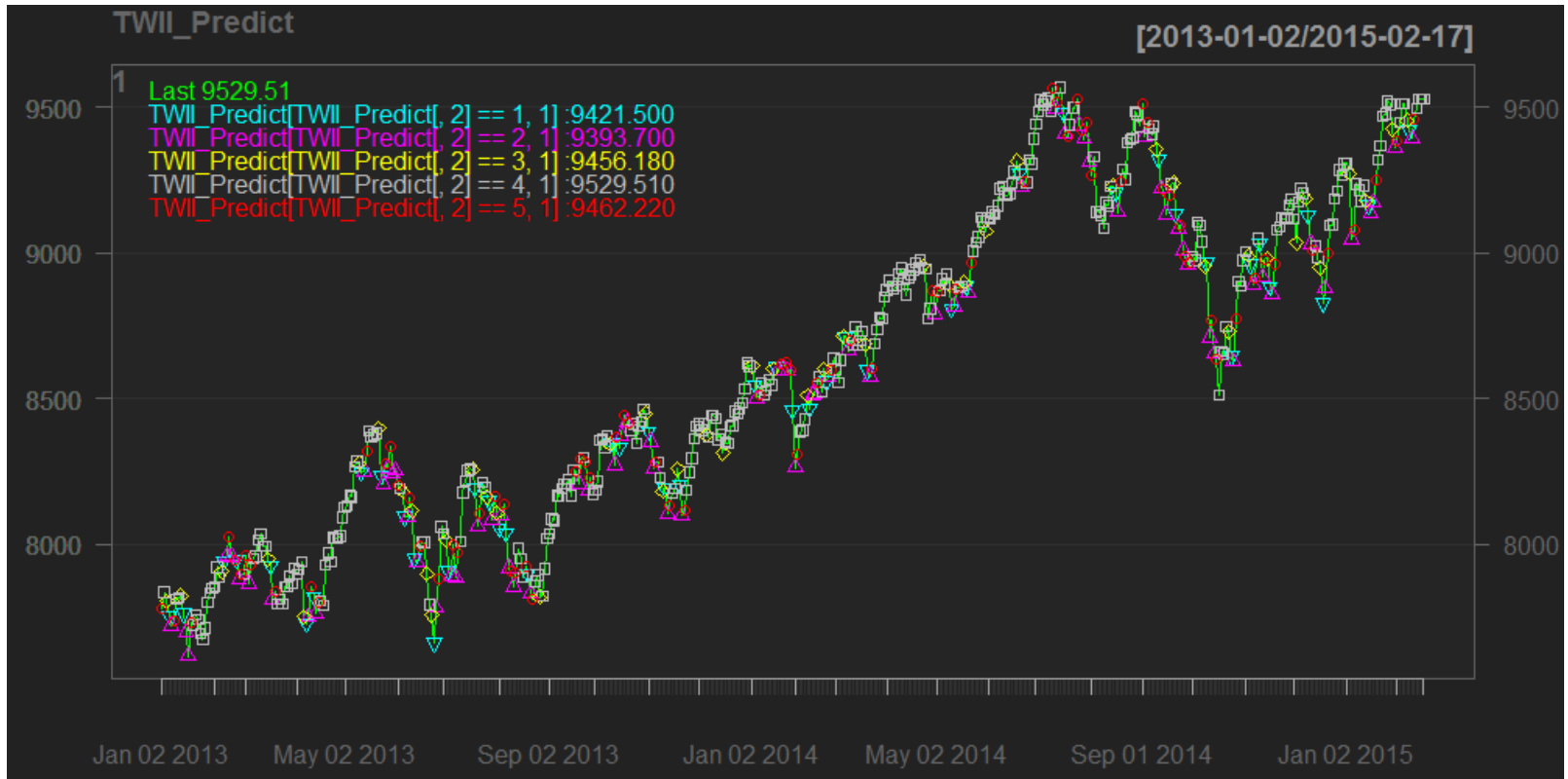
Hidden Markov Model

❖ Here is the second hidden state overlaid on top, etc...



Hidden Markov Model

❖ Here is the final graph with all of the data points represented.



Hidden Markov Model

❖ Finally, let's create a prediction of the tomorrow's stock price based upon the information learned through the Baum-Welch EM and Viterbi algorithm.

❖ Here is the R code:

```
#####  
# Here is the calculation of the next observation  
#####  
  
change <- sum(hm_model$HMM$transMat[last(VitPath$states),] *  
             .colSums((matrix(unlist(hm_model$HMM$distribution$mean),nrow=4,ncol=5))  
             * (matrix(unlist(hm_model$HMM$distribution$proportion), nrow=4,ncol=5)),  
             m=4,n=5))  
  
# select the most recent date from xts object  
  
mydata <- last(TWII_Subset)  
  
predict <- mydata$TWII.Open + change  
  
predict$TWII.Open
```

Today's Stock Market Info

row.names	TWII.Open	TWII.High	TWII.Low	TWII.Close	TWII.Adjusted
2/17/15	9500.04	9542.09	9499.97	9529.51	9529.51

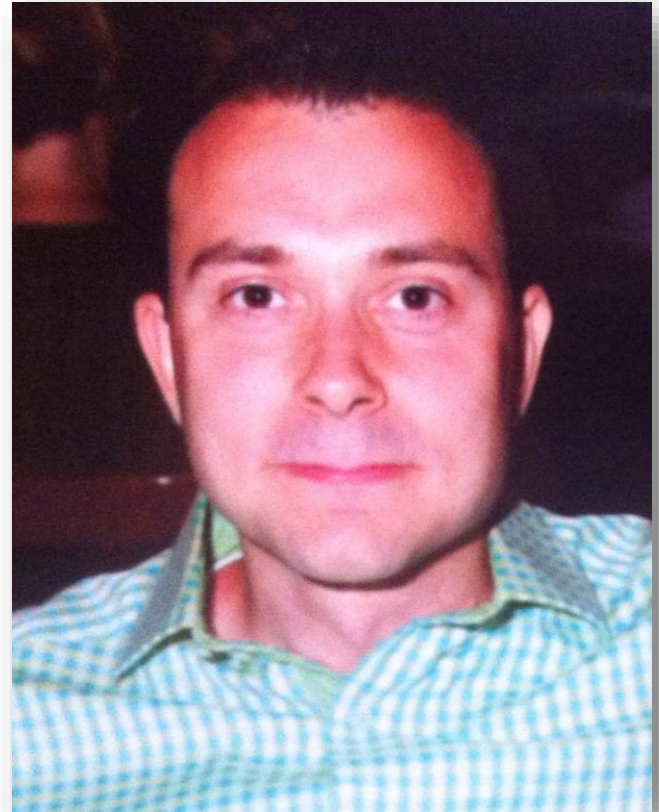


Tomorrow's Stock Prediction

Date	TWII.Open
2/18/15	9504.05

About Me

- ❖ Reside in Wayne, Illinois
- ❖ Active Semi-Professional Classical Musician (Bassoon).
- ❖ Married my wife on 10/10/10 and been together for 10 years.
- ❖ Pet Yorkshire Terrier / Toy Poodle named Brunzie.
- ❖ Pet Maine Coons' named Maximus Power and Nemesis Gul du Cat.
- ❖ Enjoy Cooking, Hiking, Cycling, Kayaking, and Astronomy.
- ❖ Self proclaimed Data Nerd and Technology Lover.



Acknowledgements

- ❖ http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html
- ❖ <https://www.robots.ox.ac.uk/~vgg/rg/slides/hmm.pdf>
- ❖ <http://www.r-bloggers.com/fun-with-r-and-hmms/>
- ❖ <http://www.biostat.jhsph.edu/bstcourse/bio638/notes/notesHMM2.pdf>
- ❖ <http://www.slideshare.net/ChiuYW/hidden-markov-model-stock-prediction>

Fine