CS 252 - Hidden Markov Models
Additional Reading 2
and
Homework problems

2 Hidden Markov Models (HMMs)

Markov chains are a simple way to model uncertainty in our computations. One thing that makes them simple is the fact that given a string, we know everything about how the model processes (or generates) it. In particular, we know which states the machine passes through, in what order, and what symbol come from what state. What if things were more complicated? What if we didn't know which state the model was in? In other words, what if the state information was hidden from us? This idea is captured in *Hidden Markov Models* (HMMs). Like Markov chains, these models are simple in the sense that they have very limited memory—state transitions are governed only by the identity of the current state and not by any earlier state history; but they are more complex (i.e., more powerful) than Markov chains because the states are hidden. What we mean by hidden is that we are not told what states the machine passes through to generate (or discriminate) a particular string. At each computational step, the machine transitions probabilistically to a state, just as in the case of Markov chains, but we don't get to observe what state. In each state, the machine probabilistically generates (or consumes) a character, which we do get to observe.

A classic example of this can be seen in music. When we listen to a musical melody, we are hearing a sequence of pitches. Behind those pitches are a sequence of chords from which the notes are generated. For certain kinds of music, it is not difficult to determine the chord progression behind the melody, but in the general case, this can be quite difficult (and there is usually not even one "right" answer). One way to model this is with an HMM. The hidden states correspond to different chords, and the generated observations are the pitches.

Another example of this occurs in natural language. Consider a sequence of words that form a sentence, where the words are atomic "alphabet" symbols and the sentence is the "string". Many words in the English language have multiple parts of speech and it can be tricky to determine what part of speech is intended for a particular word. Speech recognition and text prediction are other examples of kinds of sequences that can be modeled with HMMs.

2.1 Formal Definition of Hidden Markov Models

As you might guess, the formal definition of HMMs is quite similar to that of Markov chains.

Definition 0.0.1. A ***Hidden Markov Model*** is a 5-tuple $(Q, \Sigma, \delta, \gamma, q_0)$, where

1. Q is a finite set called the ***states***
2. Σ is a finite set called the ***alphabet***

3. $\delta : Q \cup \{q_0\} \times Q \to [0 \dots 1]$ is the ***state transition function***

4. $\gamma : Q \times \Sigma \to [0 \dots 1]$ is the ***state emission function***

5. $q_0 \notin Q$ is the ***start state***

Note the only difference from our definition for Markov chains is in the definition of γ. Now, instead of each state being associated with exactly one alphabet symbol, each state has an associated probability distribution over all the alphabet symbols. When a state is visited, it probabilistically selects one alphabet symbol to emit, based on the distribution γ. This is what makes the states unobservable. For example, in Figure 1, notice that the pitch G can be emitted in both the C state and the G state. In other words, when you hear the G note, you can not be certain with which chord it is associated—the state is hidden from direct observation.

Revisiting the Markov chain M of Figure 1, the formal definition of M is $(Q, \Sigma, \delta, \gamma, q_0)$, where

1. $Q = \{I, V, VII\}$

2. $\Sigma = \{C, C^{\#}, D, D^{\#}, E, F, F^{\#}, G, G^{\#}, A, A^{\#}, B\}$

3. δ is given as

	I	*V*	*VII*
start	0.5	0.5	0
I	0.3	0.5	0.2
V	0.3	0.3	0.4
VII	0.75	0.25	0

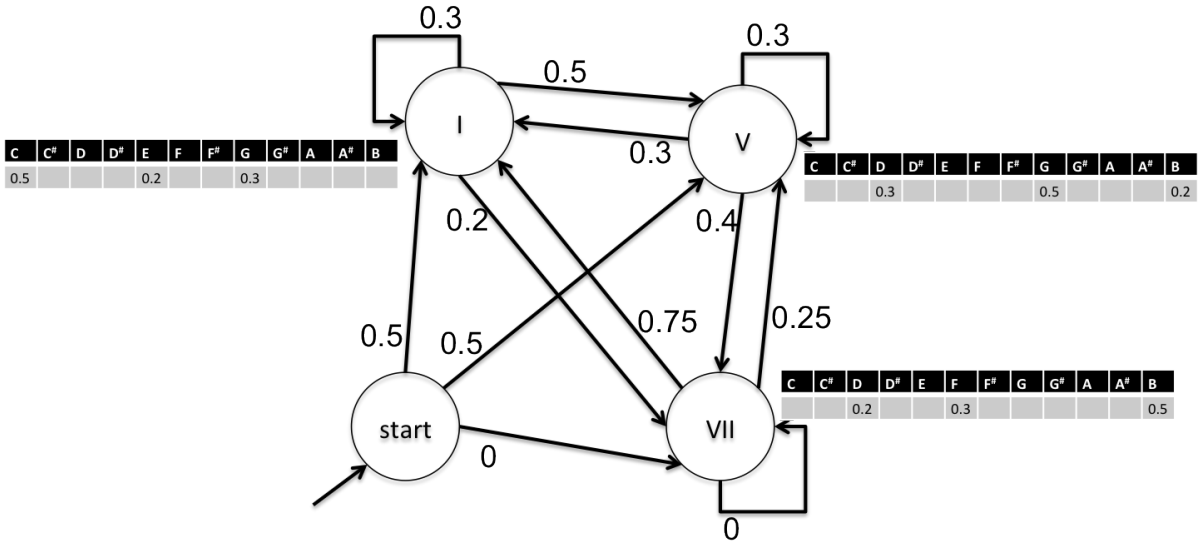

Figure 1: An HMM M that models chord progression and generates a melody in the key of C

4. γ is given as

	C	$C^{\#}$	D	$D^{\#}$	E	F	$F^{\#}$	G	$G^{\#}$	A	$A^{\#}$	B
I	0.5				0.2			0.3				
V			0.3					0.5				0.2
VII			0.2			0.3						0.5

5. $q_0 = start$

2.2 Computing with Hidden Markov Models

HMMs can be used in the same ways that Markov chains can be, though they are a bit more complicated, and they raise additional questions. We can still ask questions like "Can string w be generated by model M with probability greater than θ?" (discrimination) or "Given model M, what is the most probable string w is will generate?" (generation). But we also might ask the question, "Given that model M generates string w, what is the most likely sequence of states it went through to do so?". Because the states are hidden, the answer is not immediately obvious. But, since we already have the string, you might wonder why we care what sequence of states was used to generate it. However, it turns out that the answer to this question is where much of the power of the HMM lies—answering this question is the key to many applications like speech recognition, speech-to-text, predictive text and spell correction, spam filtering and so on. HMMs are also used to model processes in other fields as diverse as music, physics and chemistry.

2.2.1 The Viterbi algorithm

So, how do we find the most likely sequence of states for a given string? A simple, algorithm called the Viterbi algorithm is the answer. Given a machine M and a string w, the Viterbi algorithm uses an example of a process called *dynamic programming* that finds the most likely state sequence by iteratively finding the most likely next state in a greedy fashion.

Consider the machine M of Figure 1 and the pitch sequence GBGC. What sequence of state (chord progression) most likely produced these pitches? The Viterbi algorithm iterates between computing all probabilities between states given the next observation and then greedily chooses the max path to that point. When the iteration is complete, it returns the highest probability path by backtracking from the most likely final state.

The following step-by-step example illustrates. The first step, shown in Figure 2, is to compute the probability of moving from the start to each state and emitting a G from that state.

Next, we find the highest probability path to each state using all the paths available so far. In this first iteration, there are no choices since we for sure started in the start state, so this step is trivial the first time through the iteration. The second iteration next finds the probability of of moving from each state to every other state and emitting a B (see Figure 3).

Next, we find the highest probability path to each state using all the paths available so far, shown in black in Figure 4. For example, all paths to state I at the second iteration have 0 probability (all greyed out in the figure). There are two paths to state V with equal

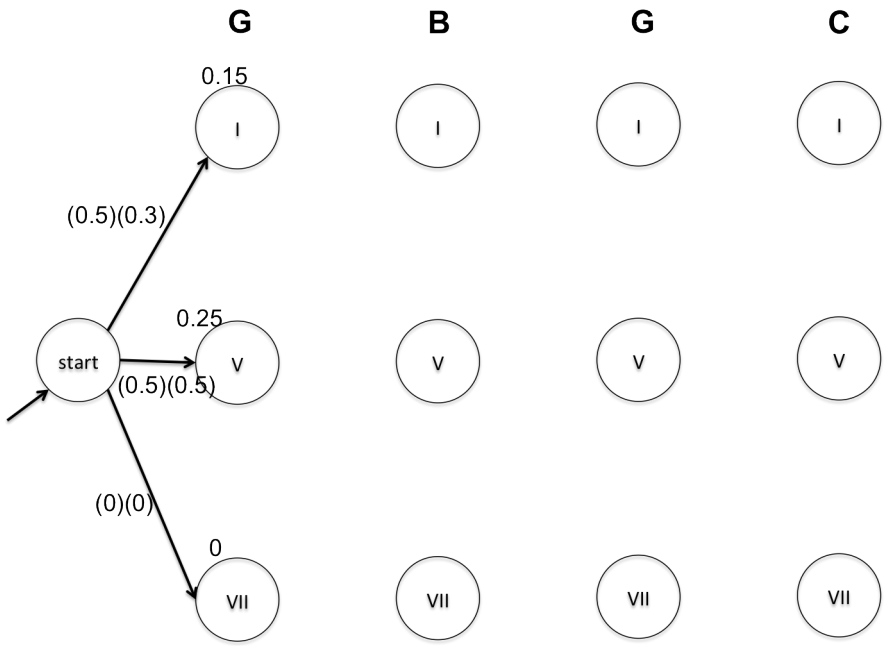

Figure 2: Viterbi step 1

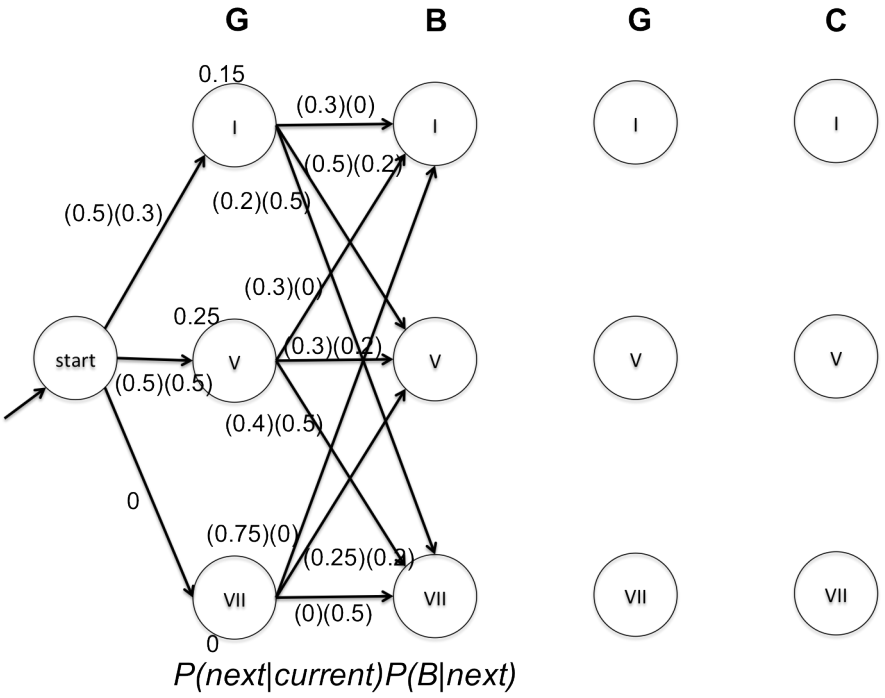

Figure 3: Viterbi step 2

probability of 0.015 (through states I and V, respectively) and one path to state V with 0 probability (through state VII, greyed out). There is one path to state VII with probability 0 (through state VII, greyed out), one path to state VII with probability 0.015 (through state I, greyed out) and one path to state VII with probability 0.05 (through state V).

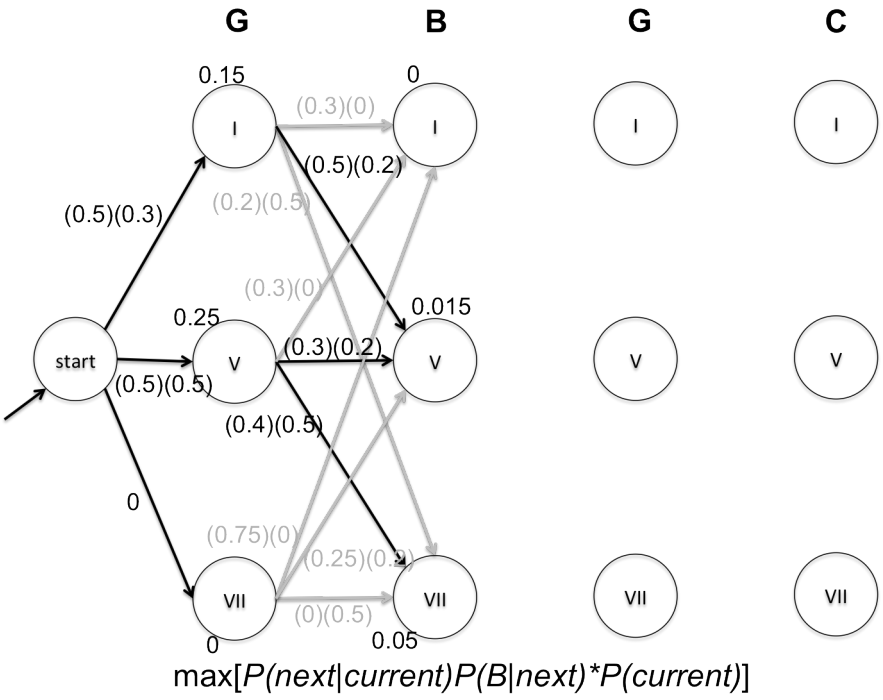

Figure 4: Viterbi step 3

Now, another interation begins and we again compute all paths from states to state, this time including the probability of emitting a G (see Figure 5).

Then, we again compute the max step (see Figure 6).

And, the process is repeated one more time, for the last pitch, C, shown in Figures 7-8.

After this last step, we see that after emitting the final C note, the model can only be in state I, and that the most probable path through the model for emitting the pitch sequence GBGC has a probability of 0.0016875. We can find the sequence of hidden states that corresponds to this probability by tracing the black arrows (which correspond to our max choices at each iteration) backwards to the start state, and this is shown in Figure 9. In this example, the most probable sequence of hidden states for producing the melody GBGC is (V, VII, I, I). If you are a musician, that means you should probably look at the chords G, B_{dim}, C and C, in order to find a good harmonization for the melody[1]. To mention some other examples, If the observations were words and the hidden states were parts-of-speech, the hidden state sequence would tell us the most probable grammatical structure of

[1]Assuming we are working in the key of C, which is what the emission matrices are designed to model. Note that changing just the emission matrices will effect transposition to another key.

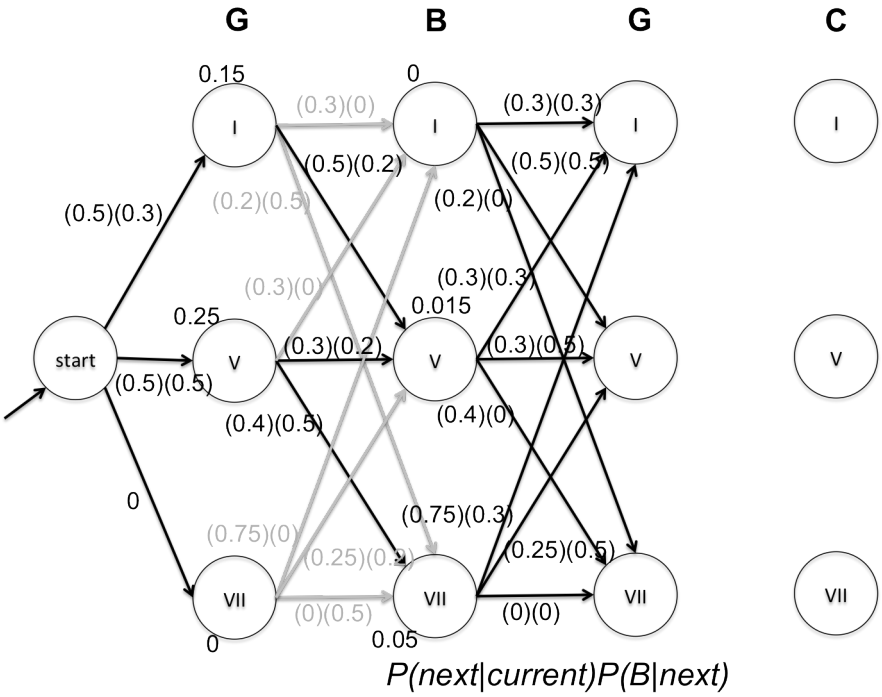

Figure 5: Viterbi step 4

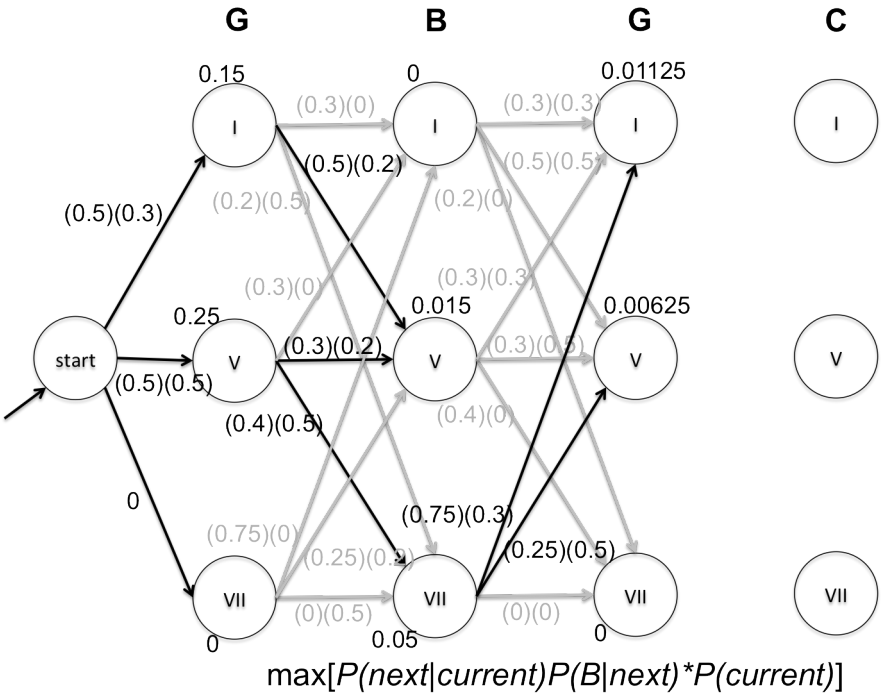

Figure 6: Viterbi step 5

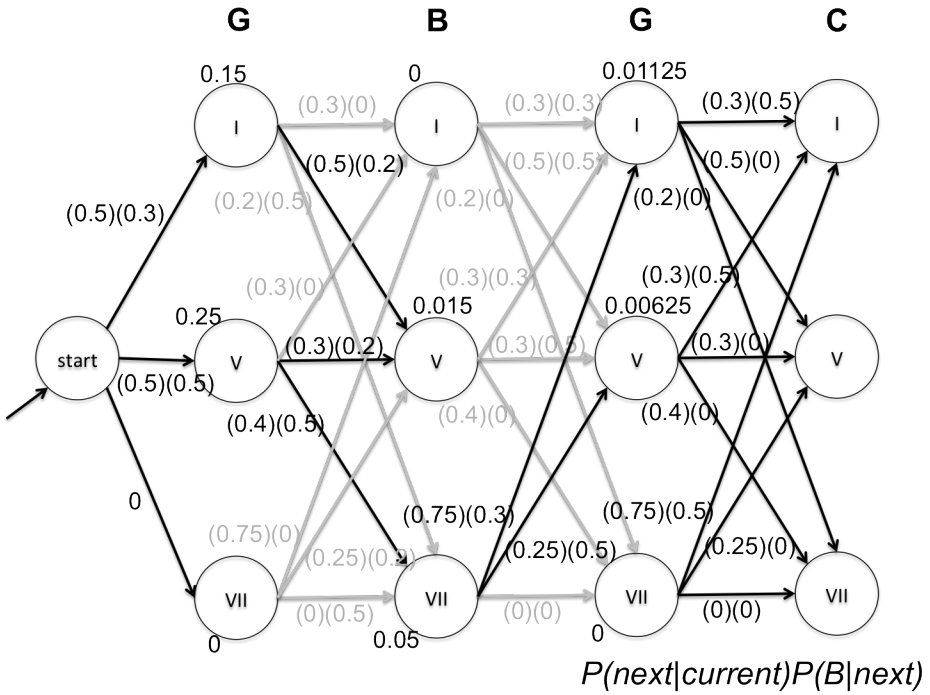

Figure 7: Viterbi step 6

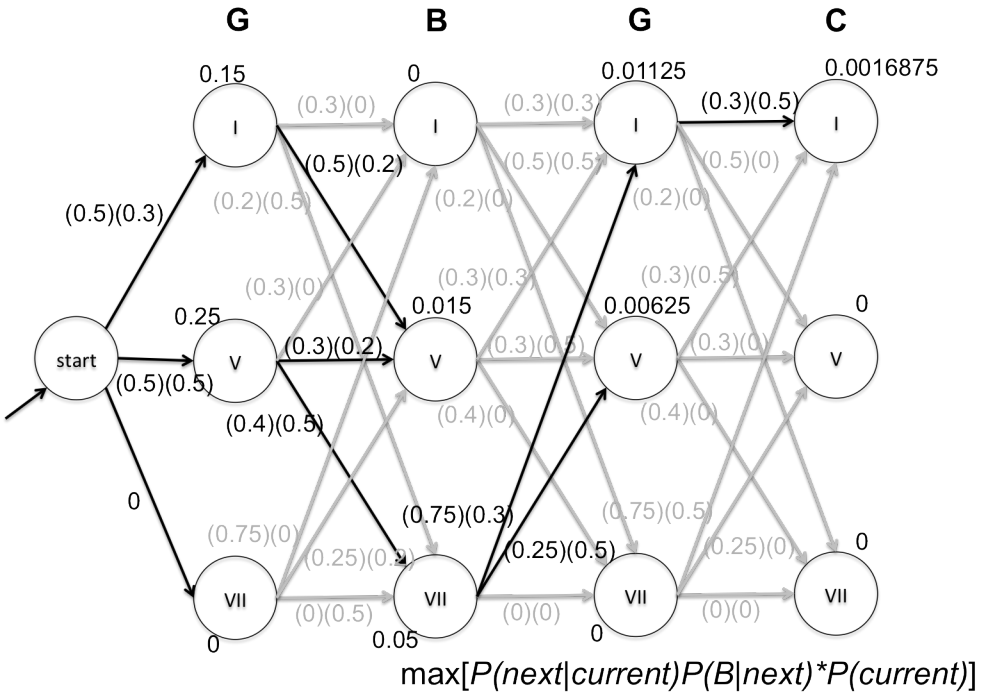

Figure 8: Viterbi step 7

a sentence. If the observations were phonemes and the hidden states were words, the hidden state sequence would tell us the most probable speech sequence.

Note that for even simple models like this one, sequences of even modest length will quickly result in probabilities so small that underflow becomes a serious concern. To ameliorate this, it is common to compute log probabilities instead. Because we are usually interested in comparative statistics, the log function will not change the outcome, and it will result in much larger numbers. In addition, because $log(xy) = log(x) + log(y)$, we get the benefit of faster computations because the multiplication of probabilities can be replaced by the addition of log probabilities[2].

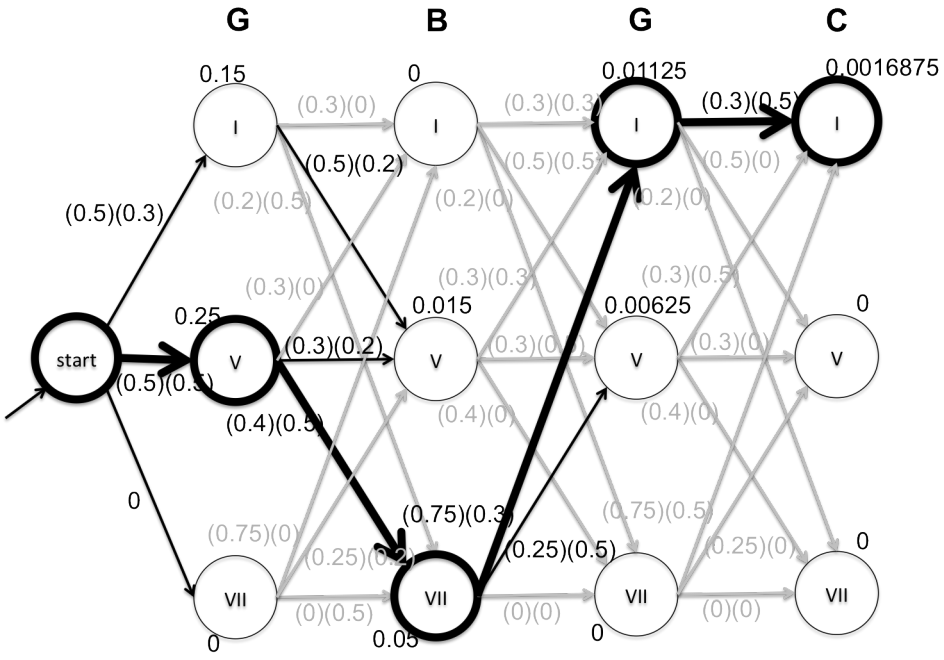

Figure 9: Viterbi step 8

2.3 Learning

Another important question, for both Markov chains and HMMs, is how to determine what the probabilistic parameters should be, and even in many cases what the states should be as well. This is an active area of research in Machine Learning that is beyond the scope of this discussion, but in the simplest cases, this parameter learning can be done by collecting statistics over example data. For instance, if you wanted to build a Markov chain for predictive text for a cell phone keyboard application, you could collect data from people typing and use that data to count occurrences of various combinations of letters. Those counts can then be converted into probabilities. If you wanted to use an HMM for speech

[2]Since $log(x) < 0$ for $0 < x < 1$, it is not uncommon to use the -log() of the probability.

recognition, you could collect recordings of various people talking under various conditions and use that data for counting statistics. The musical model above could be learned from a corpus of musical examples, from which pitch co-occurrence counts could be collected.

2.4 Exercises

Exercise 2.1. Given the following formal description of a Markov model, $M = (Q, \Sigma, \delta, \gamma, q_0)$

1. $Q = \{1, 2\}$
2. $\Sigma = \{a, b, c, d\}$
3. δ is given as

	1	*2*
start	0.6	0.4
1	0.5	0.5
2	0.3	0.7

4. γ is given as

	a	b	c	d
1	0.4	0.3	0.2	0.1
2	0.3	0.2	0.3	0.2

5. $q_0 = start$

a. Draw a transition diagram for M.

b. Use the Viterbi algorithm to compute the most probable state sequence for the string *bad*. Show your work, and report both the state sequence and its probability.