Hidden Markov Models:

Sequence Decoding using

the Viterbi Algorithm and

the Posterior Algorithm

Brian J. Walters

Description

A Markov Chain represents a series of observations in which each observation is dependent only on the previous one. A chain is constructed using a starting state and the probability of transitioning from one emission symbol to another. A Hidden Markov Model (HMM) is an extension of a Markov Chain. An HMM assumes that each observation or "emission" in a Markov Chain also depends on a hidden state. The chain relies on both the probability of an emission occurring from a hidden state and the probability of transitioning between those hidden states.

Since the states underlying the emissions are "hidden", we would like to have a way to reproduce or "decode" them. There are various decoding algorithms that attempt to recreate these hidden states. Two of these are the Viterbi algorithm and the Posterior algorithm. These algorithms use the sequence of observations, the transition probabilities between states, and the emission probabilities from the states in order to identify the hidden states that produced the sequence. The Viterbi algorithm focuses on reproducing the sequence of hidden states by determining the most probable path through the possible hidden states. The Posterior algorithm, on the other hand, determines the most probable state for each observation.

Durbin, et. al., suggest that the Viterbi algorithm will be at a disadvantage when the transition probability between states is very low (p. 61). Because Viterbi algorithm is more concerned with the overall path it will be less sensitive to the differences between particular states. The Posterior algorithm is focused on the probability of a state underlying an emission and not the entire path. This leads them to conclude that it will detect the transition between states even when the transition probability is low.

In this project, I will compare the Viterbi and Posterior algorithms and attempt to confirm Durbin's hypothesis. I will look at the two algorithms in terms of their accuracy, sensitivity, and specificity. The HMM is constructed around the data for a "sometimes dishonest casino." In this scenario, there are six possible emissions:

E = { "1", "2", "3", "4", "5", "6" }

The dice at the casino can be either in a fair or loaded state:

S = { "F", "L" }

For each dice, the emission probability is:

	1	**2**	**3**	**4**	**5**	**6**
"F"	1/6	1/6	1/6	1/6	1/6	1/6
"L"	1/10	1/10	1/10	1/10	1/10	1/2

And for each roll, the transition probability to a different die is:

	"F"	**"L"**
"F"	.95	.05
"L"	.10	.90

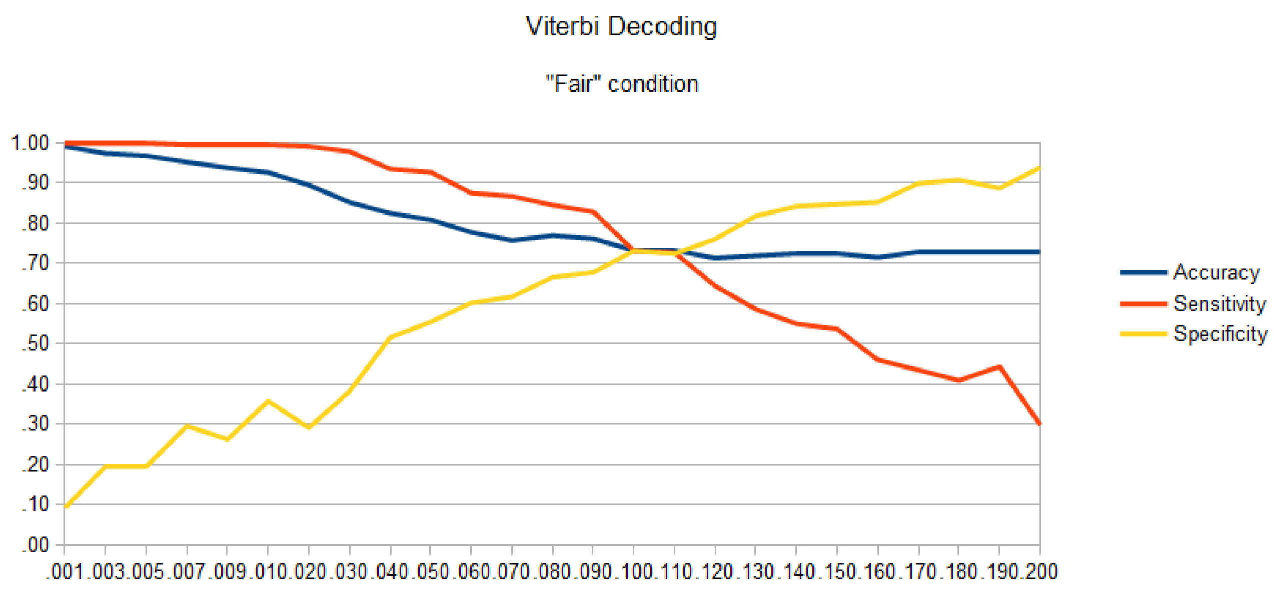
Methodology

An HMM is constructed using the “sometimes dishonest casino” data. The transition probability from “F” to “L” is altered at intervals between .001 and .2. For each, the HMM is used to construct 10 sequences of emissions and the “hidden” states that produced those emissions. The Viterbi and Posterior algorithms are then applied to produce sequences of decoded states. The accuracy, sensitivity, and specificity of the decoded sequence is calculated from the original state. Then the average accuracy, sensitivity, and specificity is calculated for the 10 trials at each interval of the altered state transitions

Data

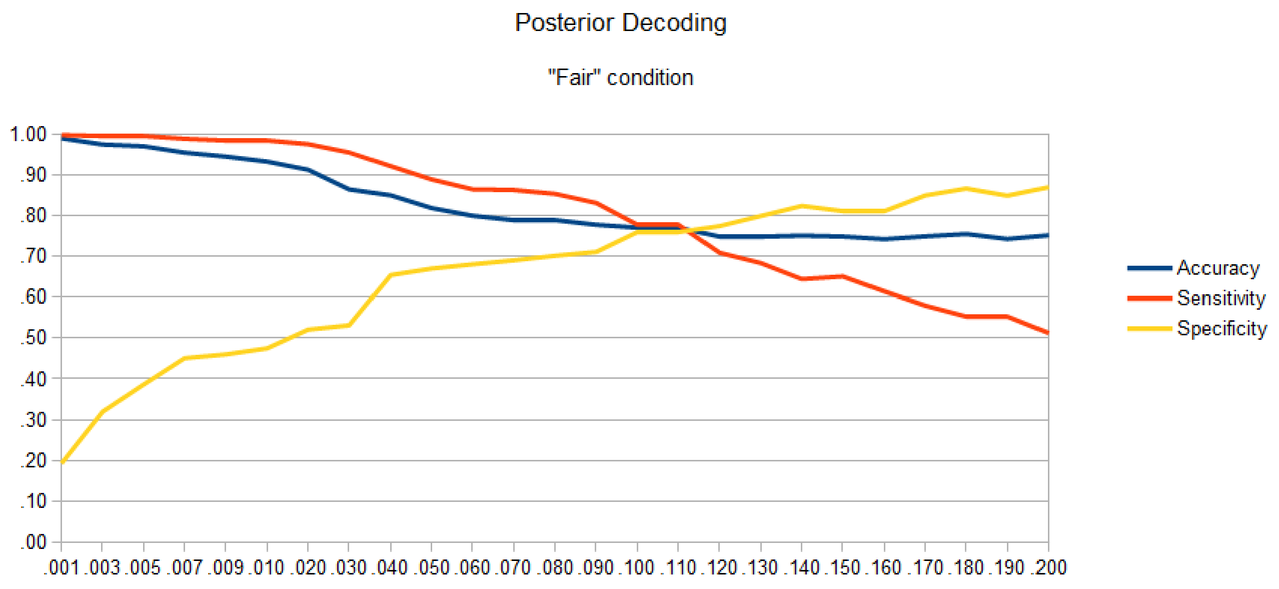
1. Viterbi Decoding

Trans. Prob.	Accuracy	Sensitivity	Specificity
.001	.9907	.9995	.0921
.003	.9736	.9990	.1947
.005	.9676	.9986	.1965
.007	.9519	.9959	.2951
.009	.9377	.9951	.2619
.010	.9263	.9939	.3574
.020	.8947	.9910	.2907
.030	.8519	.9781	.3814
.040	.8247	.9347	.5153
.050	.8081	.9268	.5544
.060	.7771	.8749	.6018
.070	.7570	.8669	.6169
.080	.7690	.8449	.6656
.090	.7613	.8283	.6776
.100	.7331	.7310	.7311
.110	.7307	.7268	.7240
.120	.7130	.6438	.7604
.130	.7189	.5858	.8179
.140	.7246	.5496	.8420
.150	.7256	.5365	.8471
.160	.7148	.4602	.8521
.170	.7266	.4345	.8989
.180	.7299	.4089	.9075
.190	.7273	.4429	.8871
.200	.7265	.2979	.9385



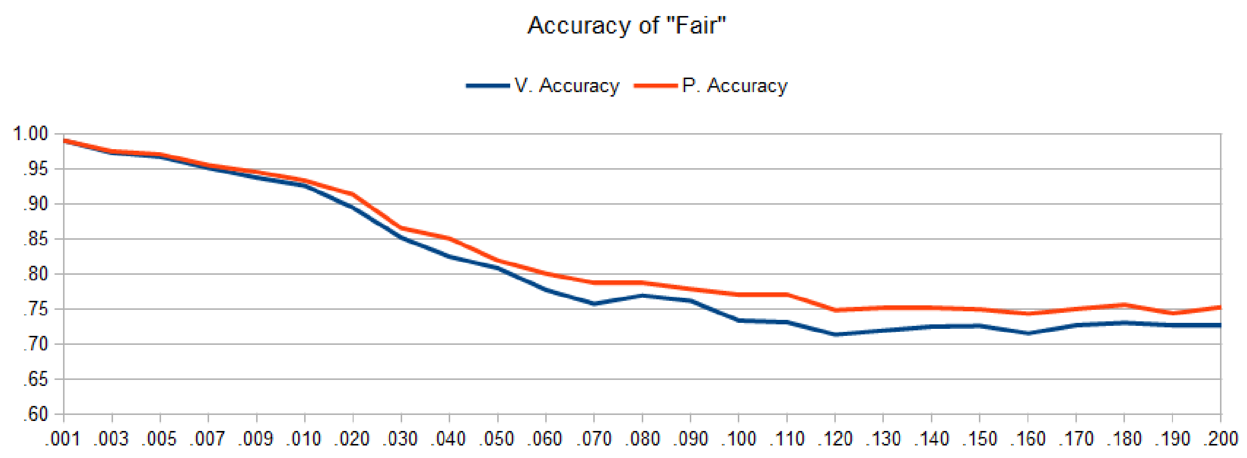
2. Posterior Decoding

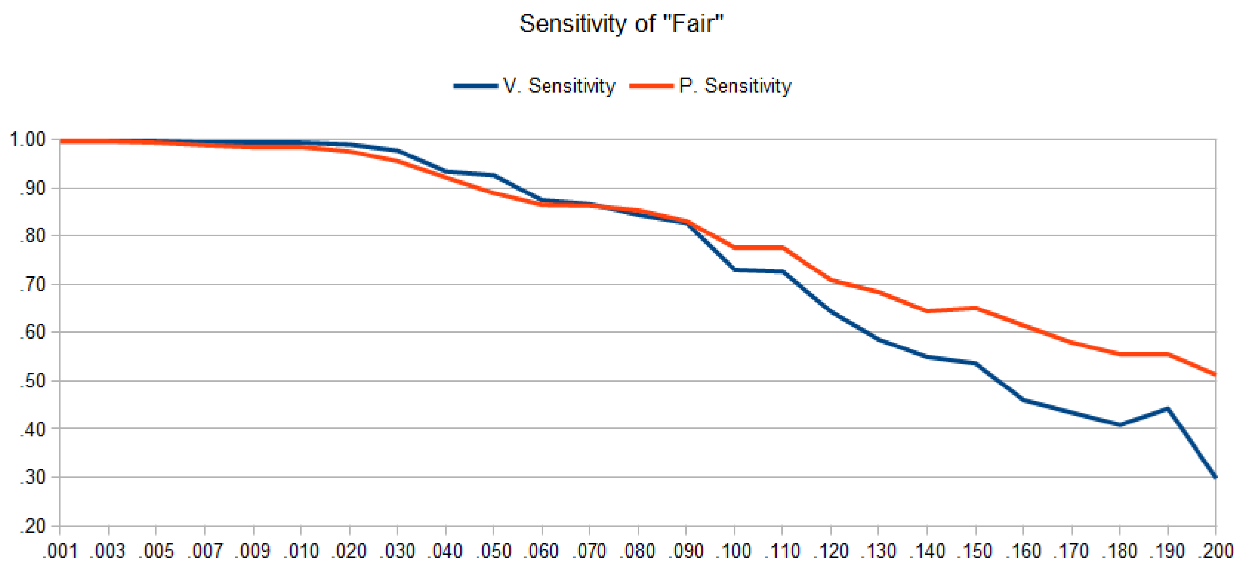
Trans. Prob.	Accuracy	Sensitivity	Specificity
.001	.9909	.9988	.1923
.003	.9757	.9968	.3193
.005	.9713	.9949	.3860
.007	.9557	.9897	.4509
.009	.9459	.9870	.4604
.010	.9336	.9868	.4751
.020	.9136	.9765	.5211
.030	.8654	.9560	.5316
.040	.8507	.9225	.6547
.050	.8192	.8898	.6706
.060	.8004	.8655	.6810
.070	.7878	.8638	.6906
.080	.7881	.8541	.7021
.090	.7782	.8319	.7114
.100	.7707	.7772	.7617
.110	.7703	.7769	.7600
.120	.7482	.7092	.7751
.130	.7509	.6843	.7998
.140	.7516	.6448	.8246
.150	.7493	.6514	.8122
.160	.7429	.6150	.8118
.170	.7499	.5798	.8504
.180	.7556	.5546	.8673
.190	.7435	.5538	.8498
.200	.7524	.5128	.8707

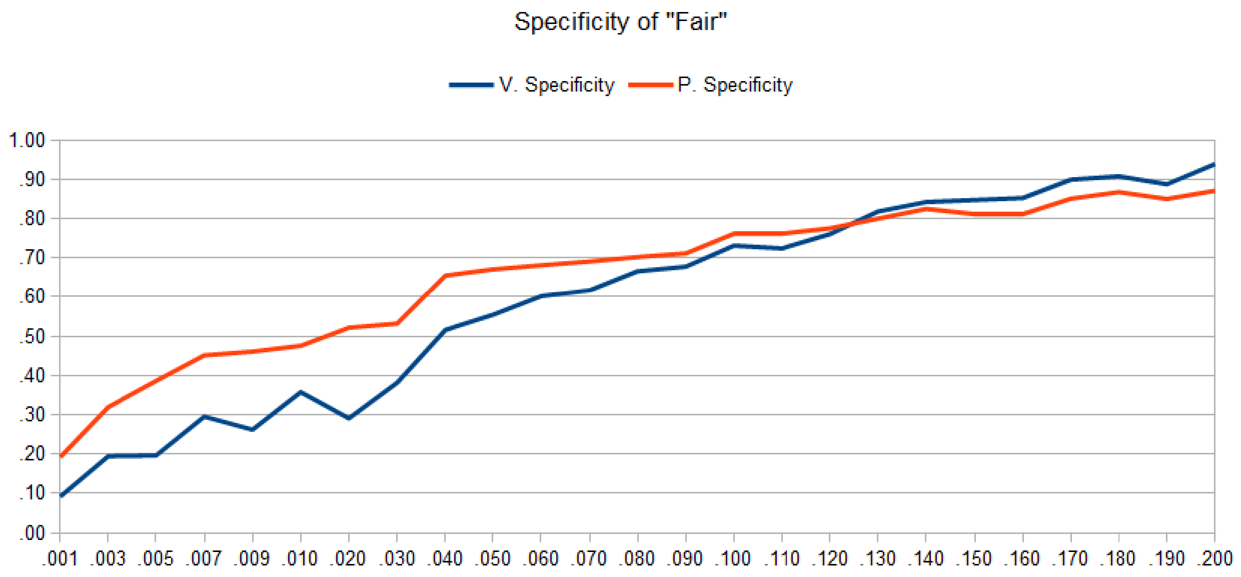


3. Comparing Viterbi and Posterior

Trans. Prob	V. Accuracy	V. Sensitivity	V. Specificity	P. Accuracy	P. Sensitivity	P. Specificity
.001	.9907	.9995	.0921	.9909	.9988	.1923
.003	.9736	.9990	.1947	.9757	.9968	.3193
.005	.9676	.9986	.1965	.9713	.9949	.3860
.007	.9519	.9959	.2951	.9557	.9897	.4509
.009	.9377	.9951	.2619	.9459	.9870	.4604
.010	.9263	.9939	.3574	.9336	.9868	.4751
.020	.8947	.9910	.2907	.9136	.9765	.5211
.030	.8519	.9781	.3814	.8654	.9560	.5316
.040	.8247	.9347	.5153	.8507	.9225	.6547
.050	.8081	.9268	.5544	.8192	.8898	.6706
.060	.7771	.8749	.6018	.8004	.8655	.6810
.070	.7570	.8669	.6169	.7878	.8638	.6906
.080	.7690	.8449	.6656	.7881	.8541	.7021
.090	.7613	.8283	.6776	.7782	.8319	.7114
.100	.7331	.7310	.7311	.7707	.7772	.7617
.110	.7307	.7268	.7240	.7703	.7769	.7600
.120	.7130	.6438	.7604	.7482	.7092	.7751
.130	.7189	.5858	.8179	.7509	.6843	.7998
.140	.7246	.5496	.8420	.7516	.6448	.8246
.150	.7256	.5365	.8471	.7493	.6514	.8122
.160	.7148	.4602	.8521	.7429	.6150	.8118
.170	.7266	.4345	.8989	.7499	.5798	.8504
.180	.7299	.4089	.9075	.7556	.5546	.8673
.190	.7273	.4429	.8871	.7435	.5538	.8498
.200	.7265	.2979	.9385	.7524	.5128	.8707




Sensitivity of "Fair"
V. Sensitivity
P. Sensitivity
1.00
.90
.80
.70
.60
.50
.40
.30
.20
.001 .003 .005 .007 .009 .010 .020 .030 .040 .050 .060 .070 .080 .090 .100 .110 .120 .130 .140 .150 .160 .170 .180 .190 .200


Specificity of "Fair"
V. Specificity
P. Specificity
1.00
.90
.80
.70
.60
.50
.40
.30
.20
.10
.00
.001 .003 .005 .007 .009 .010 .020 .030 .040 .050 .060 .070 .080 .090 .100 .110 .120 .130 .140 .150 .160 .170 .180 .190 .200

Results

The accuracy for both the Viterbi and Posterior algorithm decreases as the transition probability is increased from .001 to .2. The Viterbi algorithm's accuracy drops from .9907 to .7265 as the transition probability from fair to loaded increases. Similarly, the accuracy of the Posterior algorithm decreases from .9909 to .7524. This can be explained by the higher transition probability causing more state transitions in the original sequence. With more state transitions to detect, there is more chance for error.

In a similar fashion, both algorithms also decrease in their sensitivity to fair states as the transition probabilities increase from .001 to .2. The sensitivity of the Viterbi algorithm decreases from .9995 to . 2979 as the transition probability increases. The Posterior algorithm decreases in sensitivity from . 9988 to .5128. Both algorithms are more likely to incorrectly decode a fair state as loaded while the chance of transitions between states increases. This follows from the fact that they are more likely to identify transitions (correctly or incorrectly) as the transition probability increases.

Conversely, the specificity for both algorithms increase as the transition probability increases. The Viterbi algorithm increases in specificity from .0921 to .9385 and the Posterior algorithm from .1923 to .8707. As more loaded states appear in the original state sequence, the number of false positives becomes proportionally less significant. Thus, the overall specificity increases.

These trends, then, are expected from relatively accurate decoding algorithms. The interesting results arise from comparing the Viterbi and Posterior algorithms. They are relatively comparable in terms of accuracy up to a transition probability of .1. After this, the Posterior algorithm remains more accurate as the transition probability increases. The tradeoff, however, is that the posterior algorithm is more computationally complex.

In terms of the sensitivity to a fair condition, the Viterbi algorithm performs negligibly better while the transition probability is low. As the transition probability approach .08, however, there becomes a clear distinction in which the Posterior algorithm maintains a higher sensitivity. So, while the transition probability is low, neither algorithm is better at ruling out fair states. In fact, they are both fairly good at it. However, this is only because sequences with low transition probabilities don't include many non-fair states anyways. As the transition probability increases, the Viterbi algorithm gets worse at ruling out the fair states faster. Both will decrease because there are more non-fair states to rule out. But the Viterbi will decrease faster because of its reliance on path prediction – it is less sensitive to each individual transition as opposed to states across a sequence.

Both algorithms increase in specificity as the transition probability increases. However, the Posterior algorithm has a higher specificity when the transition probability is lower. This means that the Posterior algorithm is able to detect more of the loaded states even when there are not many of them, and there are not long sequences of them. As the transition probability reaches .12, the Viterbi algorithm begins to show a slightly higher specificity. When the transition to loaded states occurr, they are maintained longer, allowing the Viterbi algorithm to better predict that path through the states.

The data, then, confirms the claim by Durbin, et. al., that the Viterbi algorithm will have a harder time predicting state transitions when the transition probability is lower. Since the Viterbi algorithm is design to predict a path through a series of states, it performs better when states transitions, even if more frequent, are maintained longer.