1. Prove that any second-order Markov process (where the new state depends on the previous two states) can be rewritten as a first-order Markov process.

2. Consider an (unlabeled) sample of 5 instances, and two distributions $D_1$ and $D_2$ over the domain. The probabilities $D_1(x_i)$ and $D_2(x_i)$ are given by the following table. The problem is to find the $\alpha$ such that the mixture $\alpha D_1 + (1 - \alpha D_2)$ maximizes the likelihood of the sample using the Expectation-Maximization algorithm (EM).

<table>
<thead>
<tr>
<th>$i$</th>
<th>$D_1(x_i)$</th>
<th>$D_2(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Start with $\alpha = 1/2$ and show how the $\alpha$ values change over the first 10 or so iterations.

Note: this simple problem can be solved analytical by setting the derivative (wrt $\alpha$) of the likelihood, $\mathcal{L}(\alpha) = \prod_i (\alpha D_1(x_i) + (1 - \alpha) D_2(x_i))$, to zero and solving for $\alpha \in [0, 1]$. However, the goal of the problem is to use EM (but you could use the analytical solution to check your EM solution).

3. The halving algorithm makes at most $\log |C|$ mistakes when learning a concept from concept class $C$ in the noise-free, on-line learning model.

First: Show that there exists concept classes where this bound is very loose: i.e that some concept classes $C$ can be learned with far fewer than $\log |C|$ mistakes in the noise-free, on-line learning model.

Second: Describe a particular concept $C$ where the halving algorithm is not optimal: i.e. that the worst case number of mistakes made by some other algorithm is strictly less than the worst-case number of mistakes made by the halving algorithm.

Additional problems for midterm study (not to be turned in).

1. Consider the four-instance example used at the start of the boosting slides. Run AdaBoost (by hand) on that example to compute the weights of each example at each iteration (start with each example having weight $1/4$) and the weight of each of the three weak hypotheses. What is the final (un-normalized) margin of each of the four points?
2. Experiment with support vector classification in Weka on the Iris2 dataset. Use the SMO algorithm. The complexity parameter is related to the complexity per iteration, keep it at 1. Choose to standardize the data, and keep the low order terms. What degree polynomial is needed to correctly classify the training set? Do the number of support vectors grow as the degree is increased?

Use RBF (radial basis functions). How does the number of support vectors and accuracy change as a function of the gamma parameter (which controls the width of the gaussians)?

3. Can linear support vector machines (without kernels) learn the XOR function? How about polynomial kernels?

4. Run AdaBoost on the Iris2 data in Weka. How many iterations does it take to get to training error zero?

5. Calculate the probability of observation sequence (1,0,1,1,0,0) being generated, the sequence’s Viterbi (most likely) path, and the probability of the Viterbi path in the following 2-state HMM with \( \pi_1 = 1 \) and \( \pi_0 = 0. \)

\[
\begin{array}{c|cc|cc}
 s & P(s_1|s) & P(s_2|s) & P(1|s) & P(0|s) \\
\hline
 s_1 & \frac{3}{4} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\
 s_2 & \frac{1}{4} & \frac{3}{4} & \frac{1}{3} & \frac{2}{3} \\
\end{array}
\]