10-601 Machine Learning, Fall 2011: Homework 5

Machine Learning Department
Carnegie Mellon University

Due: ???????????, 5pm

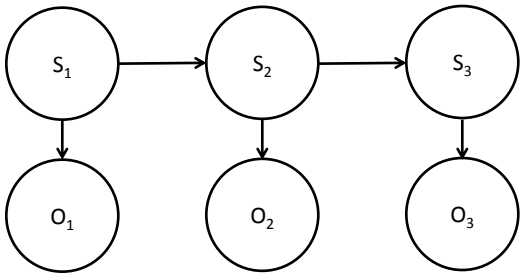
Instructions There are ???? questions on this assignment. The ?????? question involves coding, so start early. Please submit your completed homework to Sharon Cavlovich (GHC 8215) by ???????????. Submit your homework as ????? **separate** sets of pages, one for each question. Please staple or paperclip all pages from a single question together, but **DO NOT** staple pages from different questions together. This will make it much easier for the TA's to split up your homework for grading. Include your name and email address on each set.

1 Hidden Markov Models

1. Assume that we have the Hidden Markov Model (HMM) depicted in the figure above. If each of the states can take on k different values and a total of m different observations are possible (across all states), how many parameters are required to fully define this HMM? Justify your answer.

   ★ **SOLUTION:** There are a total of three probability distributions that define the HMM, the initial probability distribution, the transition probability distribution, and the emission probability distribution. There are a total of k states, so k parameters are required to define the initial probability distribution (we'll ignore all of the -1s for this problem to make things cleaner). For the transition distribution, we can transition from any one of k states to any of the k states (including staying in the same state), so k^2 parameters are required. Then, we need a total of km parameters for the emission probability distribution, since each of the k states can emit each of the m observations.
   Thus, the total number of parameters required are $k + k^2 + km$. Note that the number of parameters does not depend on the length of the HMMs.

2. What conditional independences hold in this HMM? Justify your answer.



State	$P(S_1)$
A	0.99
B	0.01

(a) Initial probs.

S_1	S_2	$P(S_2\|S_1)$
A	A	0.99
A	B	0.01
B	A	0.01
B	B	0.99

(b) Transition probs.

S	O	$P(O\|S)$
A	0	0.8
A	1	0.2
B	0	0.1
B	1	0.9

(c) Emission probs.

★ **SOLUTION:** The past is independent of the future, given the present. More precisely, we have that $S_1, ..., S_{t-1}, O_1, ...O_{t-1} \perp S_{t+1}, ..., S_n, O_{t+1}, ..., O_n | S_t$. Note that these independence relations follow directly from d-separation because a HMM is just a Bayes Net.

Suppose that we have binary states (labeled A and B) and binary observations (labeled 0 and 1) and the initial, transition, and emission probabilities as in the given table.

3. Using the forward algorithm, compute the probability that we observe the sequence $O_1 = 0$, $O_2 = 1$, and $O_3 = 0$. Show your work (i.e., show each of your alphas).

★ **SOLUTION:** The values of the different alphas and the probability of the sequence are as follows (notice that your answers may be slightly different if you kept a different number of decimal places):
$\alpha_1^A = 0.8 \times 0.99 = 0.792$
$\alpha_1^B = 0.1 \times 0.001 = 0.001$
$\alpha_2^A = 0.2(0.792(0.99) + 0.001(0.01)) = 0.156818$
$\alpha_2^B = 0.9(0.792(0.01) + 0.001(0.99)) = 0.008019$
$\alpha_3^A = 0.8(0.156818(0.99) + 0.008019(0.01)) = 0.124264$
$\alpha_3^B = 0.1(0.156818(0.01) + 0.008019(0.99)) = 0.000950699$
$P(\{O_T\}_{t=1}^T) = 0.1252147$

4. Using the backward algorithm, compute the probability that we observe the aforementioned sequence ($O_1 = 0$, $O_2 = 1$, and $O_3 = 0$). Again, show your work (i.e., show each of your betas).

★ **SOLUTION:** The values of the different betas and the probability of the sequence are as follows (again, your answers may vary slightly due to rounding):
$\beta_3^A = 1$
$\beta_3^B = 1$
$\beta_2^A = 0.99(0.8)(1) + 0.01(0.1)(1) = 0.793$
$\beta_2^B = 0.01(0.08)(1) + 0.99(0.01)(1) = 0.107$
$\beta_1^A = 0.99(0.02)(0.793) + 0.01(0.9)(0.107) = 0.157977$
$\beta_1^B = 0.01(0.2)(0.793) + 0.99(0.9)(0.107) = 0.096923$
$P(\{O_T\}_{t=1}^T) = 0.792(0.157977) + 0.001(0.096923) = 0.1252147$

5. Do your results from the forward and backward algorithm agree?

★ **SOLUTION:** Yes, the results from the forward and backward algorithm agree!

6. Using the forward-backward algorithm, compute (and report) the most likely setting for each state. Hint: you already have the alphas and betas from the above sub-problems.

★ **SOLUTION:** We already have the alphas and betas from the previous two computations. Note that the most likely state at time t is state A if $\alpha_t^A \beta_1^A > \alpha_1^B \beta_1^B$ and state B if the inequality is reversed. We predict that A is the most likely setting for every state. The relevant values for the computation are:
$\alpha_1^A \beta_1^A = 0.792 \times 0.157977 = 0.1251178$
$\alpha_1^B \beta_1^B = 0.001 \times 0.096923 = 9.6923 \times 10^{-5}$
$\alpha_2^A \beta_2^A = 0.156818 \times 0.793 = 0.1243567$
$\alpha_2^B \beta_2^B = 0.008019 \times 0.107 = 0.000858033$
$\alpha_3^A \beta_3^A = 0.124264 \times 1 = 0.124264$
$\alpha_3^B \beta_3^B = 0.000950699 \times 1 = 0.000950699$

7. Use the Viterbi algorithm to compute (and report) the most likely sequence of states. Show your work (i.e., show each of your Vs).

★ **SOLUTION:** The Viterbi algorithm predicts that the most likely sequence of states is A, A, A. The relevant computations are:
$V_1^A = 0.99 \times 0.8 = 0.792$
$V_1^B = 0.01 \times 0.1 = 0.001$
$V_2^A = 0.2(0.792)(0.99) = 0.156816$
$V_2^B = 0.9(0.792)(0.01) = 0.007128$
$V_3^A = 0.8(0.156816)(0.99) = 0.1241983$
$V_3^B = 0.1(0.007128)(0.99) = 0.000705672$

8. Is the most likely sequence of states the same as the sequence comprised of the most likely setting for each individual state? Does this make sense? Provide a 1-2 sentence justification for your answer.

★ **SOLUTION:** In this case, the two are the same. This is because the probability of changing states is so low. However, in general, the two need not be the same. In fact, we have seen an example in class where the two sequences are different.

2 Neural Networks [Mladen Kolar, 20 points]

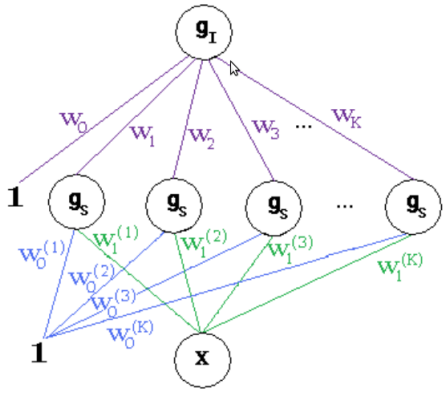
In this problem, we will consider neural networks constructed using the following two types of activation functions (instead of sigmoid functions):

- identity
$$g_I(x) = x$$
- step function
$$g_S(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

For example, the following figure represents a neural network with one input x, a single hidden layer with K units having step function activations, and a single output with identity activation. The output can be written as

$$out(x) = g_I(w_0 + \sum_{k=1}^{K} w_k g_S(w_0^{(k)} + w_1^{(k)} x))$$

Now you will construct some neural networks using these activation functions.



1. [5 pts] Consider the step function

$$u(x) = \begin{cases} y & \text{if } x \geq a, \\ 0 & \text{otherwise.} \end{cases}$$

Construct a neural network with one input x and one hidden layer, whose response is $u(x)$. Draw the structure of the neural network, specify the activation function for each unit (either g_I or g_s), and specify the values for all weights (in terms of a and y).

★ **SOLUTION:** One solution can be obtained as follows. Set $w_0 = 0, w_1 = y, w_0^{(1)} = -a, w_1^{(1)} = 1$.

2. [5 pts] Now consider the indicator function

$$\mathbb{I}_{[a,b)}(x) = \begin{cases} 1 & \text{if } x \in [a,b), \\ 0 & \text{otherwise.} \end{cases}$$

Construct a neural network with one input x and one hidden layer, whose response is $y\,\mathbb{I}_{[a,b)}(x)$, for given real values y, a and b. Draw the structure of the neural network, specify the activation function for each unit (either g_I or g_s), and specify the values for all weights (in terms of a, b and y).

★ **SOLUTION:** One solution can be obtained as follows. Set $w_0 = 0, w_1 = y, w_2 = -y, w_0^{(1)} = -a, w_1^{(1)} = 1, w_0^{(2)} = -b, w_1^{(2)} = 1$.

3. [10 points] A neural network with a single hidden layer can provide an arbitrarily close approximation to any 1-dimensional bounded smooth function. This question will guide you through the proof. Let $f(x)$ be any function whose domain is $[C, D)$, for real values $C < D$. Suppose that the function is Lipschitz continuous, that is,

$$\forall x, x' \in [C, D),\ |f(x') - f(x)| \leq L|x' - x|,$$

for some constant $L \geq 0$. Use the building blocks constructed in the previous part to construct a neural network with one hidden layer that approximates this function within $\epsilon > 0$, that is, $\forall x \in [C, D),\ |f(x) - out(x)| \leq \epsilon$, where $out(x)$ is the output of your neural network given input x. Your network should use only the activation functions g_I and g_S given above. You need to specify the number K of hidden units, the activation function for each unit, and a formula for calculating each weight w_0, w_k, $w_0^{(k)}$, and $w_1^{(k)}$, for each $k \in \{1, 2, \ldots, K\}$. These weights may be specified in terms of C, D, L and ϵ, as well as the values of $f(x)$ evaluated at a finite number of x values of your choosing (you need to explicitly specify which x values you use). You do not need to explicitly write the $out(x)$ function. Why does your network attain the given accuracy ϵ?

★ **SOLUTION:** Here is one valid solution. The hidden units all use g_s activation functions, and the output unit uses g_I. $K = \lfloor (D-C)\frac{L}{2\epsilon} + 1 \rfloor$. For $k \in \{1, \ldots, K\}$, $w_1^{(k)} = 1, w_0^{(k)} = -(C + (k-1)2\epsilon/L)$. For the second layer, we have $w_0 = 0$, $w_1 = f(C + \epsilon/L)$, and for $k \in \{2, 3, \ldots, K\}$, $w_k = f(C + (2k-1)\epsilon/L) - f(C + (2k-3)\epsilon/L)$.

We only evaluate $f(x)$ at points $C + (2k-1)\epsilon/L$, for $k \in \{1, \ldots, K\}$, which is a finite number of points.

The function value $\text{out}(x)$ is exactly $f(C + (2k-1)\epsilon/L)$ for $k \in \{1, \ldots, K\}$. Note that for any $x \in [C + (k-1)2\epsilon/L, C + k2\epsilon/l)$, $|f(x) - \text{out}(x)| \leq \epsilon$.

3 Principle Component Analysis [William Bishop, ??? points]

The fourth slide presented in class on November 10th states that PCA finds principle component vectors such that the projection onto these vectors yields minimum mean squared reconstruction error. In the case of a set of sample vectors $\vec{x}_1 \ldots, \vec{x}_n$ and finding only the *first* principle component, $\vec{v}$, this amounts to finding $\vec{v}$ which minimizes the objective:

$$J(\vec{v}) = \sum_{i=1}^{n} ||\vec{x}_i - (\vec{v}^T \vec{x}_t)\vec{v}||^2$$

To ensure that this problem has a meaningful solution we require that that $||\vec{v}|| = 1$, where $||\cdot||$ denotes the length of the vector.

In class we also learned we can view PCA as maximizing the variance of the data after it has been projected onto $\vec{v}$. If we assume the sample vectors $\vec{x}_1, \ldots, \vec{x}_n$ have zero mean, we can write this as:

$$O(\vec{v}) = \sum_{i=1}^{n} (\vec{v}^T \vec{x}_t)^2 = \vec{v}^T X X^T \vec{v}$$

where we again require $||\vec{v}|| = 1$. In the equation on there right, the columns of the matrix X are made up of the sample vectors, i.e., $X = [\vec{x}_1, \ldots, \vec{x}_n]$.

Note that to maximize $O(\vec{v})$ subject to the constraint on $\vec{v}$ we can find the stationary point of the Lagrangian:

$$L_O(\vec{v}, \lambda) = \vec{v}^T X X^T \vec{v} + \lambda(1 - \vec{v}^T \vec{v})$$

Please show that minimizing $J(\vec{v})$ can be turned into an equivalent maximization problem which has the same Lagrangian. Note that it is sufficient to show that the two Lagrangians are the same; you do not need to find the actual stationary point.

Note: $L_O(\vec{v}, \lambda)$ can equivalently be expressed as $L_O(\vec{v}, \lambda) = \left(\sum_{i=1}^{n} (\vec{v}^T \vec{x}_t)^2\right) + \lambda(1 - \vec{v}^T \vec{v})$. If you are more comfortable using this form so as to not need to deal with matrices, you may do so.

Note: For those who may need a primer on Lagrange Multipliers, appendix E of the Bishop book has some basic information. All you need to solve this problem can be found in the first few pages of this appendix.

3.1 Solution:

Answer:
Consider minimizing $J(\vec{v})$. We can write:

$$\min_{\vec{v}:||\vec{v}||=1} J(\vec{v}) = \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} ||\vec{x}_i - (\vec{v}^T \vec{x}_t)\vec{v}||^2 \right)$$

In the above notation $\min_{\vec{v}:||\vec{v}||=1} J(\vec{v})$ means that we minimize $J(\vec{v})$ *such that* the length of $\vec{v}$ is 1. Continuing our derivation:

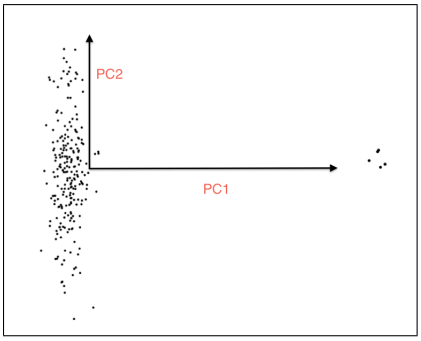
$$\begin{aligned}
\min_{\vec{v}:||\vec{v}||=1} J(\vec{v}) &= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} ||\vec{x}_i - (\vec{v}^T \vec{x}_i)\vec{v}||^2 \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} [\vec{x}_i - (\vec{v}^T \vec{x}_i)\vec{v}]^T [\vec{x}_i - (\vec{v}^T \vec{x}_i)\vec{v}] \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} [\vec{x}_i^T - (\vec{v}^T \vec{x}_i)\vec{v}^T][\vec{x}_i - (\vec{v}^T \vec{x}_i)\vec{v}] \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{x}_i^T \vec{x}_i - (\vec{v}^T \vec{x}_i)\vec{v}^T \vec{x}_i - \vec{x}_i^T (\vec{v}^T \vec{x}_i)\vec{v} + (\vec{v}^T \vec{x}_i)\vec{v}^T (\vec{v}^T \vec{x}_i)\vec{v} \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{x}_i^T \vec{x}_i - \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} - \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} + (\vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v})\vec{v}^T \vec{v} \right)
\end{aligned}$$

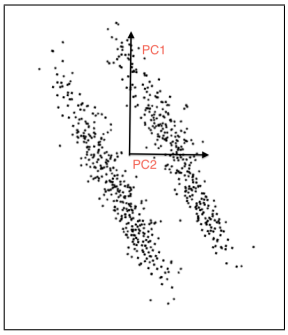
In the last line we have used the fact that $(\vec{v}^T \vec{x}_i)\vec{v}^T \vec{x}_i = \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v}$, $\vec{x}_i^T (\vec{v}^T \vec{x}_i)\vec{v} = \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v}$ and $(\vec{v}^T \vec{x}_i)\vec{v}^T (\vec{v}^T \vec{x}_i)\vec{v} = (\vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v})\vec{v}^T \vec{v}$

We can then write:

$$\begin{aligned}
\min_{\vec{v}:||\vec{v}||=1} J(\vec{v}) &= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{x}_i^T \vec{x}_i - \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} - \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} + (\vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v})\vec{v}^T \vec{v} \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{x}_i^T \vec{x}_i + \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v}(-2 + \vec{v}^T \vec{v}) \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v}(-2 + \vec{v}^T \vec{v}) \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v}(-2 + 1) \right) \\
&= \min_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} -\vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} \right) \\
&= \max_{\vec{v}:||\vec{v}||=1} \left(\sum_{i=1}^{n} \vec{v}^T \vec{x}_i \vec{x}_i^T \vec{v} \right) \\
&= \max_{\vec{v}:||\vec{v}||=1} \left(\vec{v}^T X X^T \vec{v} \right)
\end{aligned}$$

2. [5 pts] Draw the first two principle components for the following two datasets.





Based on this, comment on a limitation of PCA.

Answer: One of the limitations of PCA is that is can be highly affected by just a few outliers. Another limitation is that it may fail to capture meaningful components that do not pass through the origin.