10-601 Machine Learning: Homework 6

Due 5:30 p.m. Thursday, April 14, 2016
TAs: Maria De Arteaga and Renato Negrinho

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that. You *must* turn in at least $n-1$ of the n homeworks to pass the class, even if for zero credit.
- **Collaboration policy:** Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Submission:** You must submit your solutions on time BOTH electronically by submitting to autolab AND by dropping off a hardcopy in the bin outside Gates 8221 by 5:30 p.m. Thursday, April 14, 2016. We recommend that you use LaTeX, but we will accept scanned solutions as well. On the Homework 6 autolab page, you can click on the "download handout" link to download the submission template, which is a tar archive containing a blank placeholder pdf for your written solutions, and an Octave `.m` file for each programming question. Replace each of these files with your solutions for the corresponding problem, create a new tar archive of the top-level directory, and submit your archived solutions online by clicking the "Submit File" button.

  DO NOT change the name of any of the files or folders in the submission template. In other words, your submitted files should have exactly the same names as those in the submission template. Do not modify the directory structure.

Problem 1: Fundamentals of Bayesian Networks (Maria) [35 pts]

Figure 1 shows a Bayesian Network that encodes the dependencies between the following variables: Season (S), Temperature (T), Clear sky (C), Rain (R), and Park usage (PU).

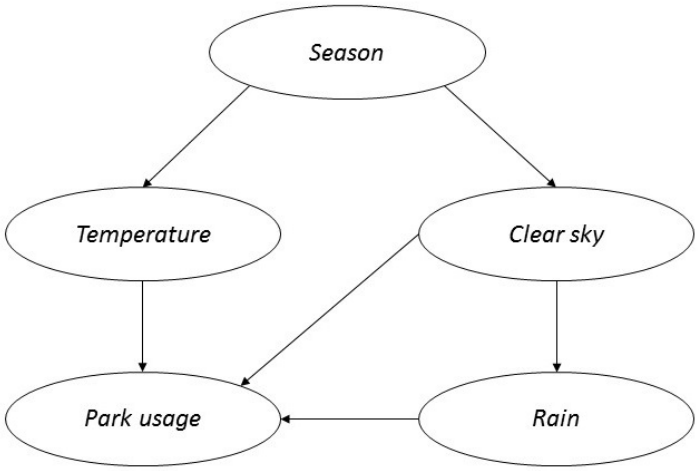

Figure 1: Bayesian Network that represents the joint distribution over the variables used to predict park usage.

Independence and d-separation [16 pts]

(a) [10 pts] Using the graphical model depicted in Figure 1 state whether the following independence statements are True or False.

1. $S \perp R$
2. $S \perp PU \mid T$
3. $T \perp C \mid S$
4. $S \perp PU \mid T, R$
5. $T \perp C \mid PU$
6. $S \perp PU \mid R$
7. $S \perp PU \mid T, C$
8. $S \perp PU \mid T, C, R$
9. $S \perp PU \mid C$
10. $T \perp R \mid S$

(b) [6 pts] D-separation. For the following questions provide your answer and a brief justification.

1. Which variables are d-separated from R?
2. Which variables are d-separated from R given S?
3. Which variable(s) do you need to condition on so that T and R are d-separated? Is there only one way of achieving this d-separation?

Factorization of joint distributions [10 pts]

(c) [5 pts] Write down the factorized form of the joint distribution over all the variables represented in Figure 1, $P(S, T, C, R, PU)$.

(d) [5 pts] Draw a minimal Bayesian network that represents the following factorization of the joint distribution:

$$P(S, T, C, R, PU) = P(S)P(C)P(PU|T, R)P(R \mid S, C)P(T \mid S) \tag{1}$$

Probability queries [9 pts]

$p(\textbf{season} = summer)$	$p(\textbf{season} = winter)$
0.4	0.6

season	$p(\textbf{temp} = warm \mid \textbf{season})$	$p(\textbf{temp} = cold \mid \textbf{season})$
summer	0.8	0.2
winter	0.05	0.95

season	$p(\textbf{clear sky} = Yes \mid \textbf{season})$	$p(\textbf{clear sky} = No \mid \textbf{season})$
summer	0.7	0.3
winter	0.45	0.55

clear sky	$p(\textbf{rain} = Yes \mid \textbf{clear sky})$	$p(\textbf{rain} = No \mid \textbf{clear sky})$
yes	0.01	0.99
no	0.6	0.4

temperature	**clear sky**	**rain**	$p(\textbf{park usage} = Yes \mid \ldots)$	$p(\textbf{park usage} = No \mid \ldots)$
warm	Yes	Yes	0.2	0.8
warm	Yes	No	0.95	0.05
warm	No	Yes	0.05	0.95
warm	No	No	0.7	0.3
cold	Yes	Yes	0.5	0.5
cold	Yes	No	0.6	0.4
cold	No	Yes	0.01	0.99
cold	No	No	0.2	0.8

Table 1: Conditional probability tables for the Bayesian Network shown in Figure 1.

(e) [9 pts] In this exercise you will evaluate probability queries using the information provided in Table 1. Evaluate each of the probability queries listed below and show your calculations.

1. What is the probability that the sky is clear, given it is summer?
2. What is the probability that people are using parks, given it is raining and it is summer?
3. What is the probability that people are using parks, given the sky is clear?

Problem 2: Homophily or contagion, why did Ian drink beer? (Maria) [15 pts]

Ian and Joey are friends, but Ian's parents are concerned about the influence Joey might be having over Ian. Joey and Ian were together one day and Joey was drinking beer. The next day Ian tried beer for the first time. Ian's parents believe he tried beer *because* Joey drank it before.

His parents think this is a case of social contagion. But can we be certain that this is the cause? Or is it possible that they became friends over their interest in yeast, and this lead both of them to be interested in trying beer (latent homophily on an unobserved characteristic)? In this exercise you will explore the answer to these questions.

(a) [5 pts] As a first step, you will draw a Bayesian network that can represent this setting. We define the following variables:

- $Y_{i,t}$: response variable for individual i at time t (if she/he drank beer at time t)
- Z_i: observed traits that don't change over time
- X_i: unobserved traits that don't change over time
- $A_{i,j}$: binary variable indicating whether i considers j to be its friend (note that this is a directed edge).

We consider a model where where we have two individuals, Y_i and Y_j (imagine Ian is individual i and Joey is individual j). All the following hold:

- For each individual i, j, both the observed traits and unobserved traits of the individual directly influence its response at both time $t-1$ and at time t
- For each i and j the response variable at time $t-1$ directly influences the response variable at time t (recall the response variable of i at time t is $Y_{i,t}$)
- The network tie $A_{i,j}$ is influenced by the unobserved traits of both individual i and individual j
- The response of individual i at time t can be directly influenced by the response of individual j at time $t-1$ if $A_{i,j} = 1$.

Assuming $A_{i,j} = 1$ and using all of the above statements, draw the edges in the Bayesian Network shown in Figure 2.

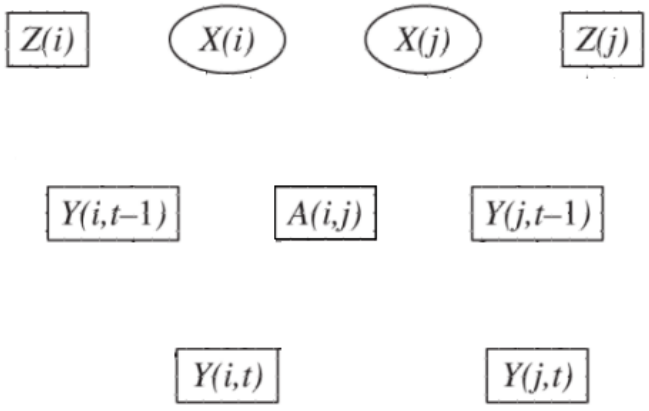

Figure 2: Fill in missing edges to obtain Bayesian Network that follows the listed conditions.

(b) [4 pts] If we see a change in $Y_{j,t-1}$ and then we see a change in $Y_{i,t}$, we want to know if we can guarantee that this is due to the social link between them (social contagion). Phrase this question in terms of d-separation.

(c) [6 pts] If we see a change in $Y_{j,t-1}$ and then we see a change in $Y_{i,t}$, can we guarantee that this is due to the social link between them (social contagion)? Justify.

Problem 3: Hidden Markov Models (Renato) [50 pts]

A HMM defines a joint probability distribution over sequences of state-observation pairs. Let $\mathcal{S} = \{1, \ldots, n\}$ denote the set of n possible states and $\mathcal{O} = \{1, \ldots, m\}$ denote the set of m possible observations. Furthermore, define two special states: *start* and *stop*. Every state sequence begins with *start* and ends with *stop*. There are no observations associated with these states, and they only occur, respectively, at the beginning and the end of state sequences.

Note that the states *start* and *stop* are normal states in much the same way as the states in $\mathcal{S}$, but because it is impossible to transition back into start and to transition out of stop, it is convenient to single them out in this way.

A HMM is determined by two types of parameters: state transition parameters $\theta \in \mathbb{R}^{n \times n}$, $\theta_{start} \in \mathbb{R}^n$, and $\theta_{stop} \in \mathbb{R}^n$, where $\theta_{ij} = P_{S|S'}(j|i)$, $\theta_{start,j} = P_{S|S'}(j|start)$, and $\theta_{i,stop} = P_{S|S'}(stop|i)$; state observation parameters $\gamma \in \mathbb{R}^{n \times m}$, where $\gamma_{ij} = P_{O|S}(j|i)$. Furthermore, we have the normalization constraints $\sum_{j=1}^{n} \theta_{ij} + \theta_{i,stop} = 1$, $\sum_{j=1}^{n} \theta_{start,j} = 1$ for the state transition parameters and $\sum_{j=1}^{n} \gamma_{ij} = 1$ for the state observation parameters, for $i \in \mathcal{S}$. Note that for each state $i \in \mathcal{S}$, we have a categorical distribution with parameters $\theta_{i1}, \ldots, \theta_{in}, \theta_{i,stop}$ for the transitions out of state i, and a categorical distribution with parameters $\gamma_{i1}, \ldots, \gamma_{im}$ for the observations in state i. There is also a categorical distributions with parameters $\theta_{start,1}, \ldots, \theta_{start,n}$ for the initial transition out of the state *start*. We assume that empty sequences are impossible (i.e., $T = 0$). The state transition and state observation parameters are the same across all time steps and all sequences. It is said that the HMM is homogeneous.

Given the above definition of a HMM, a sequence of state-observation pairs of length T has probability

$$P_{S_{0:T+1},O_{1:T}}(start, s_{1:T}, stop, o_{1:T}; \theta, \theta_{start}, \theta_{stop}, \gamma) = \prod_{t=1}^{T+1} P_{S|S'}(s_t|s_{t-1}; \theta, \theta_{start}, \theta_{stop}) \prod_{t=1}^{T} P_{O|S}(o_t|s_t; \gamma), \quad (2)$$

where $s_0 = start$, $s_{T+1} = stop$, $s_t \in \mathcal{S}$, and $o_t \in \mathcal{O}$, for $t \in \{1, \ldots, T\}$. From now on, we will use a simplified notation where we omit the parameters and subscripts when the meaning is clear from context.

Structure [8 pts]

(a) [2 pts] Draw the graphical model corresponding to HMM model for a sequence of state-observation pairs with length $T = 3$. Don't forget to include the random variables associated with the start and stop states.

(b) [2 pts] A generative model, as it is the case of a HMM, has a generative story that describes how to sample from the model. What is the generative story for the HMM described above?

(c) [2 pts] List two types of conditional independence statements that can be read of the HMM graphical model structure.

(d) [2 pts] Explain what is the Markov blanket of a node in a Bayesian network in terms of the notion of d-separation and active paths. What is the Markov blanket for state nodes? What about for observation nodes?

Estimation [8 pts]

We will now derive maximum likelihood estimator for the HMM parameters given a set of fully observed N training examples $\mathcal{D} = \{(s_{1:T_1}^{(1)}, o_{1:T_1}^{(1)}), \ldots, (s_{1:T_N}^{(N)}, o_{1:T_N}^{(N)})\}$, with lengths $T_1, \ldots, T_N$.

It is convenient to remember that for a categorical random variable X over K different events with parameters $\beta_1, \ldots, \beta_K$ with $\sum_{k=1}^{K} \beta_k = 1$, the probability of event $x \in \{1, \ldots, K\}$ can be written as

$$P_X(x) = \prod_{k=1}^{K} \beta_k^{\mathbb{1}[x=k]},$$

where $\mathbb{1}[\cdot]$ is the indicator function. This can be similar to what you have done in previous homeworks to write the likelihood under a Bernoulli model, which is just a categorical distribution with $K = 2$.

(e) [2 pts] Write the log likelihood $\ell(\theta, \theta_{start}, \theta_{stop}, \gamma)$ for the training set $\mathcal{D}$.

(f) [6 pts] Show that the log likelihood $\ell(\theta, \theta_{start}, \theta_{stop}, \gamma)$ can be written as a separable function of the parameters θ_{start}, and $\theta_{i,:}, \theta_{i,stop}, \gamma_{i,:}$ for all $i \in \mathcal{S}$, i.e., $\ell(\theta, \theta_{start}, \theta_{stop}, \gamma) = \ell(\theta_{start}) + \sum_{i=1}^{n} \ell(\theta_{i,:}) + \ell(\theta_{i,:}, \theta_{i,stop}) + \ell(\gamma_{i,:})$. What are the statistics of the training data needed to evaluate the log likelihood?

The maximum likelihood solution for the log-likelihood cannot just be computed by differentiating and equating to zero. This is because of the normalization constraints for the distributions, i.e., $\sum_{j=1}^{m} \theta_{start,j} = 1$, and $\sum_{j=1}^{n} \theta_{i,j} + \theta_{i,stop} = 1$, $\sum_{j=1}^{m} \gamma_{i,j} = 1$, for all $i \in \mathcal{S}$. The problem can be solved by introducing Lagrange multipliers for the constraints. We get the following natural estimators:

$$\begin{aligned}
\hat{\theta}_{ij} &= \frac{c_{\text{trans}}(i,j)}{c(i)}, \\
\hat{\theta}_{start,j} &= \frac{c_{\text{trans}}(start,j)}{N}, \\
\hat{\theta}_{i,stop} &= \frac{c_{\text{trans}}(i,stop)}{c(i)}, \\
\hat{\gamma}_{ij} &= \frac{c_{\text{obs}}(i,j)}{c(i)},
\end{aligned}$$

where

$$\begin{aligned}
c_{\text{trans}}(i,j) &= \sum_{k=1}^{N} \sum_{t=1}^{T_k - 1} \mathbb{1}[s_t^{(k)} = i, s_{t+1}^{(k)} = j], \\
c_{\text{trans}}(i,stop) &= \sum_{k=1}^{N} \mathbb{1}[s_{T_k}^{(k)} = i, s_{T_k+1}^{(k)} = stop], \\
c_{\text{trans}}(start,j) &= \sum_{k=1}^{N} \mathbb{1}[s_0^{(k)} = start, s_1^{(k)} = j], \\
c(i) &= \sum_{j=1}^{n} c_{\text{trans}}(i,j) + c_{\text{trans}}(i,stop) \\
&= \sum_{k=1}^{N} \sum_{t=1}^{T_k} \mathbb{1}[s_t^{(k)} = i] \\
c_{\text{obs}}(i,j) &= \sum_{k=1}^{N} \sum_{t=1}^{T_k} \mathbb{1}[s_t^{(k)} = i, o_t^{(k)} = j].
\end{aligned}$$

Estimating the parameters of a HMM from labelled data reduces to counting and normalizing.

Inference [12 pts]

Given a fixed HMM, i.e., a HMM with fixed parameters, θ, θ_{start}, θ_{stop}, and γ, there are various queries that we may want to answer. For example, we could want to know what is the probability of a sequence of observations $o_{1:T}$, i.e, $P_{O_{1:T}}(o_{1:T})$. Evaluating this requires marginalizing over all possible sequences of states that may have generated $o_{1:T}$, i.e.,

$$P_{O_{1:T}}(o_{1:T}) = \sum_{s_{1:T} \in \mathcal{S}^T} P_{S_{0:T+1}, O_{1:T}}(start, s_{1:T}, stop, o_{1:T}). \tag{3}$$

Doing the summation naively is intractable. An efficient approach will exploit the HMM structure. A more common query is given a sequence of observations $o_{1:T}$, what is the most likely sequence of states $s_{1:T}$ that gave rise to $o_{1:T}$. This is called MAP or Viterbi decoding and it is written as

$$\hat{s}_{1:T} = \underset{s_{1:T}}{\operatorname{argmax}} P_{S_{0:T+1}|O_{1:T}}(start, s_{1:T}, stop|o_{1:T}). \tag{4}$$

Another common way of decoding predicts the most likely state for each position in the sequence, given the sequence of observations $o_{1:T}$ and marginalizing out all the other state random variables. This is called marginal decoding and it is written as

$$\hat{s}_t = \underset{s_t}{\operatorname{argmax}} P_{S_t|O_{1:T}}(s_t|o_{1:T}), \tag{5}$$

for $t = 1, \ldots, T$. The marginal over S_t can be computed naively as

$$P_{S_t|O_{1:T}}(s_t|o_{1:T}) = \sum_{s_{-t} \in \mathcal{S}^{T-1}} P_{S_{0:T+1}|O_{1:T}}(start, s_{1:T}, stop|o_{1:T}),$$

where s_{-t} denotes a state assignment to all state variables except S_t.

(g) [2 pts] Show that the decoding rule 4 is equivalent to $\operatorname{argmax}_{s_{1:T}} P(start, s_{1:T}, stop, o_{1:T})$.

(h) [2 pts] What is the time complexity of doing Viterbi decoding naively in terms of the number of states n, the number of observations m, and the length of the sequence T (i.e. evaluating $\hat{s}_{1:T} = \operatorname{argmax}_{s_{1:T}} P(start, s_{1:T}, stop, o_{1:T})$)?

We will now show how the factorization 3 can be used to derive an efficient algorithm for Viterbi decoding. The derivation is similar to the one for Forward-Backward algorithm seen in class.

(i) [6 pts] Derive an efficient algorithm to do Viterbi decoding. Hint: Start with the equivalent definition given in (g), substitute the definition of the joint, use the fact that not all terms in the product depend on all the variables. Consider defining the following recursion $\alpha_1(j) = P_{S|S'}(j|start)P_{O|S}(o_1|j)$, $\alpha_{t+1}(j) = \max_{i \in \mathcal{S}} \alpha_t(i)P_{S|S'}(j|i)P_{O|S}(o_{t+1}|j)$, for $t = 1, \ldots, T-1$, and $\alpha_{T+1} = \max_{i \in \mathcal{S}} \alpha_T(i)P_{S|S'}(stop|i)$. The term $\alpha_t(j)$ can be interpreted as the maximum score for a partial state assignment ending at state j at step t.

(j) [2 pts] What is the computational complexity of Viterbi decoding using the algorithm derived in the previous exercise?

Implementation [18 pts]

We will now train a HMM for a Natural Language Processing task: Named Entity Recognition (NER). In what follows we interchangeably talk of states as tags or labels and observations as words. We will learn the parameters from labelled data, implement Viterbi decoding and compare it to a simple baseline that does independent predictions for each label in the sequence.

Named Entity Recognition is a sequence labelling task that seeks to identify elements in text from specific categories. The labels have a BIO specifiers (*begin, inside*, and *outside*). In our case, there are four categories: *Person, Organization, Location* and *Miscellaneous*. There are nine labels total: eight for the cross-product of the four categories with the *begin* and *inside* specifiers; one for the *outside* specifier. An example sentence from the dataset is shown below:

B-LOC	O	B-PER	O	B-LOC	O	B-PER	O
Iraq	's	Saddam	meets	Russia	's	Zhirinovsky	.

There are structural constraints in the tagging scheme: a label of type *inside* has to be preceded by a label of type *begin* of the same category.

A NER dataset from the CoNNL 2003 shared task has been provided in *data.mat*. Both the train and test datasets have been preprocessed, and both tags and words have been indexed and substituted by integers. The file *data.mat* contains:

- *train*: Structure with the training data.
  - *word_seqs*: Cell array with the word sequences.
  - *tag_seqs*: Cell array with the tag sequences. Matching dimensions to *word_seqs*.
- *test* Test dataset. Same structure as the training dataset.
- *index_to_word*: Contains the mapping from integers to words used to preprocess the data. Words that were not in the vocabulary were mapped to *OOV*. Can be used with the function *map_to_readable* to get back the (preprocessed) readable sequences.
- *index_to_tag*: Same as *index_to_word*, but for tags. For tags there is no *OOV* equivalent.

For the implementation questions, you will be asked to complete parts of the code provided. The naming convention used in the code follows in part the notation in the writeup. We briefly describe the structure of the code provided.

- *baseline_train.m*: Contains the function `[baseline_params] = baseline_train(state_seqs, obs_seqs, n, m)`. Used to train a model that does independent prediction for each of the labels just based on the observation of that position. The statistics that you will need to collect here are the same as the observation statistics for the HMM.
- *baseline_decode.m*: Contains the function `[pred_state_seqs] = baseline_decode(baseline_params, obs_seqs)` You will need to do baseline decoding using the parameters computed in `baseline_train`, i.e., for each word in each of the sentences, predict the tag that occurred most frequently with that word.
- *hmm_train.m*: Contains the function `[hmm_params] = hmm_train(state_seqs, obs_seqs, n, m, alpha_obs, alpha_trans`. You will have to collect the statistics from the training data that are necessary to evaluate the estimators for the parameters of the HMM.
- *hmm_decode.m*: Contains the function `[pred_state_seqs] = hmm_decode(hmm_params, obs_seqs)`. You will have to implement the Viterbi decoding here.
- *map_to_readable.m*: Used to map sequences back to a readable format.
- *main.m*: Code for training the model and running all the experiments. After the functions have been completed, these can be used to obtain test and train results.

(k) [3 pts] Complete the function `[baseline_params] = baseline_train(state_seqs, obs_seqs, n, m)`. Compute the co-occurrence counts of state-observation pairs.

(l) [2 pts] Complete the function `[pred_state_seqs] = baseline_decode(baseline_params, obs_seqs)`. For each observation, you will predict the label that occurred most frequently with it.

(m) [5 pts] Complete the function `[hmm_params] = hmm_train(state_seqs, obs_seqs, n, m, alpha_obs, alpha_trans)` . You will need to collect the statistics from the training data according to the expressions computed in the estimation section. The arguments `alpha_obs` and `alpha_trans` are parameters for add-k smoothing for the state observation probabilities and the state transition probabilities.

(n) [8 pts] Complete the function `[pred_state_seqs] = hmm_decode(hmm_params, obs_seqs)`. You will need to implement the computation of the Viterbi messages and the backpointers to recover the most probable label sequence. Note: you will work in log-probabilities, rather than directly with probabilities, to avoid numeric underflow. This does not change the maximum probability labelling of the sequence.

Analysis [4 pts]

(o) [2 pts] Run the code with `alpha_obs = 0.1` and `alpha_trans = 0`. What are the results that you get? How do you justify the difference in accuracy between the baseline and Viterbi decoding?

(p) [2 pts] Run the code with `alpha_obs = 0` and `alpha_trans = 0`. Are the results for Viterbi better or worse that in the previous exercise? How do you justify the difference?

Submission Instructions

To submit your work through Autolab, put the following files in a folder named *hw6*, compress it, and submit the resulting tar file: *hw6.tar*. Please, do not include the data file *data.mat* in the tar file, as this will exceed the current submission size limit.

- *baseline_train.m*
- *baseline_decode.m*
- *hmm_train.m*
- *hmm_decode.m*
- *written_solutions.pdf*