Homework 7: Graphical Models; Active Learning

10-601B: Machine Learning (Fall 2016)
Out November 22, 2016
Due 5:30 p.m. Monday, December 5, 2016

TAs: Fish Tung, Tianshu Ren, Sriram Vasudevan

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.
- **Submission:** You must submit your code for problem 3 on time electronically by submitting to autolab by 5:30 p.m. Monday, December 5, 2016. On the Homework 7 autolab page, you can click on the "download handout" link to download the submission template, which is a tar archive containing a 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.

  For write-up questions, submit your written answers on Gradescope.
- **Collaboration policy**: Please read the policy on the course webpage: http://www.cs.cmu.edu/~mgormley/courses/10601b-f16/about.html

Problem 1: Disagreement-based Active Learning [30 pts]

T or F [12 pts]

Answer the following questions. If your answer is **T**, provide 1-2 sentences of justification. If your answer is **F**, give a counter-example if possible. Recall from the lecture notes that a classification problem is **realizable** if the true labeling function h^* is in our hypothesis class H. Given a set of labeled examples $\{(x_t, y_t)\}_{t=1}^n$, where $y_t = h^*(x_t), \forall t = 1, \dots, n$, the version space $H' \subseteq H$ is a subset of H in which the hypotheses are consistent with the labels seen so far. The region of disagreement with respect to a version space H', denoted as $\text{DIS}(H')$, is defined to be the part of data space about which there is still some uncertainty. More specifically, $\text{DIS}(H') = \{x \in X : \exists h_1, h_2 \in H', h_1(x) \neq h_2(x)\}$.

(a) Assume in the agnostic case, given a concept class H and a sequence of training instances $\{(x_t, y_t)\}_{t=1}^n$. Let $H_t \subseteq H$ be the version space after the learner has received the t-th instance. Then $H_t \neq \varnothing$, $\forall t = 1, \dots, n$.

Solution:

(b) Assume the problem is realizable. Given a concept class H and a sequence of training instances $\{(x_t, y_t)\}_{t=1}^n$. Let $H_t \subseteq H$ be the version space after the learner has received the t-th instance. After all n examples have been observed we have $|H_n| \geq 1$.

Solution:

(c) Let $X = \{(x_t, y_t)\}_{t=1}^n$ be a set of n training instances and H be the hypothesis space. Assume our current version space is H'. Pick $x_0 \in \text{DIS}(H')$. Let $\hat{H}$ be the version space of X after querying x_0, i.e., the version space of $X \cup \{(x_0, y_0)\}$, where y_0 is provided by the true labeling function, then it must be that $\hat{H} \subset H'$.

Solution:

An Example [18 pts; 3pts for each question]

Consider a binary classification problem where the sample space is $\Omega = \{(\mathbf{x}, y) : x \in \mathbb{R}, y \in \{+1, -1\}\}$. Let the concept class $H = \{(a, b), a \leq b\}$, where the decision rule is given by:

$$h_{(\mathbf{a},\mathbf{b})}(\mathbf{x}) = \begin{cases} +1 & a < x < b, \\ -1 & otherwise \end{cases}$$

In this problem we assume that H is realizable. Suppose after receiving a sequence of training instances (2.0, +1), (2.5, +1), (-1.0, -1), (0.0, -1) (4.0, -1).

(d) Draw the given sample and explicitly write down the version space $H_1 \subseteq H$ that is consistent with all the instances.

Solution:

(e) Write down the region of disagreement for space H_1, i.e, DIS(H_1).

Solution:

(f) Suppose we receive a new instance (0.5, +1). Does this sample lie in the disagreement region DIS(H_1)? Write down the new version space $H_2 \subseteq H$ that is consistent with all the instances including the new instance.

Solution:

(g) This time another new instance (3.0, -1). Does this sample lie in the disagreement region DIS(H_2)? Write down the new version space $H_3 \subseteq H$ that is consistent with all the instances including the two new instances.

Solution:

(h) Write down the region of disagreement for space H_3, i.e, DIS(H_3).

Solution:

(i) Comparing H_1, H_2 and H_2, H_3. Does it satisfy the answer you have in Problem 1(c)?

Problem 2: Bayesian Network [20 pts]

Figure 1 shows a Bayesian Network that encodes the dependencies between the following variables: Course (C), Assignment (A), Exam (E), Interview (I), Sleep (S) and Healthy (H).

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

1. $A \perp E$
   Solution:
2. $C \perp H$
   Solution:

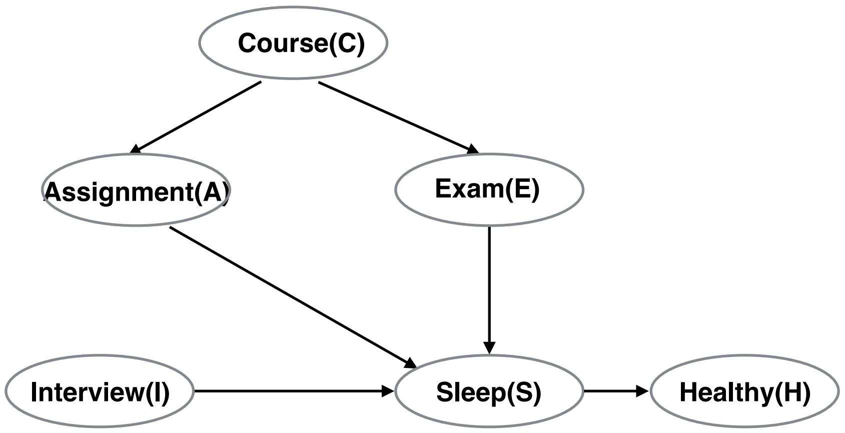

Figure 1: Bayesian Network that represents the joint distribution over the variables used to predict park usage.

3. $I \perp C$
   Solution:
4. $I \perp H$
   Solution:
5. $C \perp H \mid S$
   Solution:
6. $I \perp C \mid S$
   Solution:
7. $A \perp E \mid C$
   Solution:
8. $A \perp H \mid C, E$
   Solution:
9. $S \perp C \mid E, I, A$
   Solution:
10. $A \perp E \perp I \perp H \mid C, S$
    Solution:

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

1. Which variables are d-separated from H?
   Solution:
2. Which variables are d-separated from A given C?
   Solution:
3. Which variable(s) do you need to condition on so that C and H are d-separated?
   Solution:

(c) [4 pts] Write down the factorized form of the joint distribution over all the variables represented in Figure 1, $P(C, A, E, I, S, H)$.

Solution:

Problem 3: Hidden Markov Models [50 pts]

A HMM defines a joint probability distribution over sequences of state-observation pairs. Let $\mathcal{Y} = \{1, \dots, n, n+1, n+2\}$ denote the set of $n+2$ possible states and $\mathcal{X} = \{1, \dots, m\}$ denote the set of m possible observations. Note that state $n+1$ is define to be the state "start", and state $n+2$ is defined to be the state of "stop." For simplicity, we define two constant START $= n+1$ and STOP $= n+2$.

A HMM is determined by two types of parameters: emission parameters $A \in \mathbb{R}^{n \times m}$ and state transition parameters $B \in \mathbb{R}^{(n+2)\times(n+2)}$. For a sequence of length T, for $k = \{1, 2, \cdots, T\}$, $A_{ij} = P(X_k = j | Y_k = i)$ and $B_{ij} = P(Y_k = j | Y_{k-1} = i)$. Assume $P(Y_0) =$ START with probability 1 and $P(Y_{T+1}) =$ STOP with probability 1 at both train and test time. The transition parameter from the initial state to time step 1 is the $(n+1)$-th row of B, i.e., $B_{\text{START},j} = P(Y_1 = j | Y_0 = \text{START})$ and the transition from T to $T+1$ is the $(n+2)$-th column of B, i.e., $B_{i,\text{STOP}} = P(Y_{T+1} = \text{STOP} | Y_T = i) \in \mathbb{R}^n$ 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

$$P(Y_{0:T+1}, X_{1:T}; B, A) = P(Y_0) \prod_{t=1}^{T+1} P(Y_t | Y_{t-1}; B) \prod_{t=1}^{T} P(X_t | Y_t; A), \tag{1}$$

where $Y_t \in \mathcal{Y}$ for $t \in \{0, \dots, T+1\}$, and $X_t \in \mathcal{X}$, for $t \in \{1, \dots, T\}$. 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 $T = 3$.

Solution:

(b) [3 pts] Briefly describe how samples are generated from HMM.

Solution:

(c) [3 pts] List two types of conditional independence statements from the HMM graphical model structure.

Solution:

Estimation [8 pts]

We will now derive maximum likelihood estimator for the HMM parameters given a set of fully observed N training examples $\mathcal{D} = \{(y^{(1)}_{1:T_1}, x^{(1)}_{1:T_1}), \dots, (y^{(N)}_{1:T_N}, x^{(N)}_{1:T_N})\}$, with lengths $T_1, \dots, T_N$. Note that using the assumption that $P(Y_0) =$ START with probability 1 and $P(Y_{T+1}) =$ STOP with probability 1 at both train and test time, $y^{(k)}_0 = START$ and $y^{(k)}_{T+1} = STOP$ for all examples in $\mathcal{D}$.

It is convenient to remember that for a categorical random variable X over K different events with parameters $\beta_1, \dots, \beta_K$ with $\sum_{k=1}^{K} \beta_k = 1$, the probability of event $x \in \{1, \dots, K\}$ 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 $K = 2$.

(d) [2 pts] Write the log likelihood $\ell(B, A)$ for the training set $\mathcal{D}$.

Solution:

(e) [6 pts] Show that the log likelihood $\ell(B, A)$ can be written as a separable function of the parameters $B_{\mathrm{START},j}$, and $B_{i,:}, B_{i,\mathrm{STOP}}, A_{i,:}$ for all $i \in \mathcal{Y}$, i.e., $\ell(B, A) = \ell(B_{\mathrm{START},j}) + \sum_{i=1}^{n} \ell(B_{i,:}) + \ell(B_{i,:}, B_{i,\mathrm{STOP}}) + \ell(A_{i,:})$. What are the statistics of the training data needed to evaluate the log likelihood?

Solution:

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., $\sum_{j=1}^{n} B_{\mathrm{START},j} = 1$, and $\sum_{j=1}^{|\mathcal{Y}|} B_{i,j} = 1$, and $\sum_{j=1}^{m} A_{i,j} = 1$, for all $i \in \mathcal{X}$. The problem can be solved by introducing Lagrange multipliers for the constraints. We get the following natural estimators:

$$\hat{B}_{ij} = \frac{c_{\mathrm{trans}}(i,j)}{c(i)},$$
$$\hat{A}_{i,j} = \frac{c_{\mathrm{obs}}(i,j)}{c(i)},$$

where

$$c_{\mathrm{trans}}(i,j) = \sum_{k=1}^{N} \sum_{t=1}^{T_k} \mathbb{1}[y_t^{(k)} = i, y_{t+1}^{(k)} = j],$$
$$c(i) = \sum_{j=1}^{|\mathcal{Y}|} c_{\mathrm{trans}}(i,j)$$
$$c_{\mathrm{obs}}(i,j) = \sum_{k=1}^{N} \sum_{t=1}^{T_k} \mathbb{1}[y_t^{(k)} = i, x_t^{(k)} = j].$$

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, B and A, 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 $X_{1:T}$, i.e, $P(X_{1:T})$. Evaluating this requires marginalizing over all possible sequences of states that may have generated $x_{1:T}$, i.e.,

$$P(X_{1:T} = x_{1:T}) = \sum_{Y_{1:T} \in \mathcal{Y}^T} P(Y_{1:T}, X_{1:T} = x_{1:T}). \tag{2}$$

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

$$\hat{Y}_{1:T} = \operatorname*{argmax}_{Y_{1:T}} P(Y_{1:T} | X_{1:T} = x_{1:T}). \tag{3}$$

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

$$\hat{Y}_t = \operatorname*{argmax}_{Y_t} P(Y_t | X_{1:T} = x_{1:T}), \tag{4}$$

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

$$P(Y_t | X_{1:T} = x_{1:T}) = \sum_{Y_{-t} \in \mathcal{Y}^{T-1}} P(Y_{1:T} | X_{1:T} = x_{1:T}),$$

where Y_{-t} denotes a state assignment to all state variables except Y_t.

(f) [2 pts] Show that the decoding rule 3 is equivalent to $\text{argmax}_{Y_{1:T}} P(Y_{1:T}, X_{1:T} = x_{1:T})$.

Solution:

(g) [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 $\hat{Y}_{1:T} = \text{argmax}_{Y_{1:T}} P(Y_{1:T}, X_{1:T} = x_{1:T})$)?

Solution:

We will now show how the factorization 2 can be used to derive an efficient algorithm for Viterbi decoding. The derivation is similar to the one for Forward-Backward algorithm found in the Bishop textbook.

(h) [6 pts] Derive an efficient algorithm to do Viterbi decoding. Hint: Start with the equivalent definition given in (f), 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 $\alpha_0(\text{START}) = 1$, $\alpha_0(i) = 0$ for $\forall i \neq \text{START}$, and $\alpha_{t+1}(j) = \max_{i \in \mathcal{Y}} \alpha_t(i) P(Y_{t+1} = j | Y_t = i) P(X_{t+1} = x_{t+1} | Y_{t+1} = j)$, for $t = 1, \ldots, T-1$. The term $\alpha_t(j)$ can be interpreted as the maximum score for a partial state assignment ending at state j at step t.

Solution:

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

Solution:

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.

(j) [1 pts] What is j = $\mathrm{argmax}_j\, B_{\mathrm{START},j}$, where B is the MLE? (example answer: the least frequent words.)

(k) [1 pts] What is i = $\mathrm{argmax}_i\, B_{i,\mathrm{STOP}}$, where B is the MLE?

(l) [1 pts] What is $B_{\mathrm{STOP},j}$, where B is the MLE and $j \neq \mathrm{STOP}$?

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

(n) [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.

(o) [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.

(p) [5 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]

(q) [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?

Solution:

(r) [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?

Solution:

Submission Instructions

To submit your work through Autolab, put the following files in a folder named *hw7*, compress it, and submit the resulting tar file: *hw7.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*