Hidden Markov Models [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

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

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 $\mathcal{D} = \{(s^{(1)}_{1:T_1}, o^{(1)}_{1:T_1}), \ldots, (s^{(N)}_{1:T_N}, o^{(N)}_{1:T_N})\}$, 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.,

$$P_{O_{1:T}}(o_{1:T}) = \sum_{s_{1:T} \in \mathcal{S}^T} P_{S_{0:T+1}, O_{1:T}}(start, s_{1:T}, stop, o_{1:T}). \tag{2}$$

Doing the summation naively is intractable. An efficient approach will exploit the HMM structure. A more common query is given a sequence of observations $o_{1:T}$, 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

$$\hat{s}_{1:T} = \underset{s_{1:T}}{\operatorname{argmax}} P_{S_{0:T+1}|O_{1:T}}(start, s_{1:T}, stop|o_{1:T}). \tag{3}$$

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

$$\hat{s}_t = \underset{s_t}{\operatorname{argmax}} P_{S_t|O_{1:T}}(s_t|o_{1:T}), \tag{4}$$

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 3 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 2 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?