LING/COMP 445, LING 645
Problem Set 6

Name: , **McGill ID**:
Collaborators:

Due before 4:35 PM on Wednesday, December 13, 2023

Please enter your name and McGill ID above. There are several types of questions below.

- For questions involving answers in English or mathematics or a combination of the two, put your answers to the question in an **Answer** section like in the example below. You can find more information about LaTeX here `https://www.latex-project.org/`.
- For programming questions, please put your answers into a file called `ps6-lastname-firstname.clj`. Be careful to follow the instructions exactly and be sure that all of your function definitions use the precise names, number of inputs and input types, and output types as requested in each question.

  For the code portion of the assignment, **it is crucial to submit a standalone file that runs**. Before you submit `ps6-lastname-firstname.clj`, make sure that your code executes correctly without any errors when run at the command line by typing `clojure ps6-lastname-firtname.clj` at a terminal prompt. We cannot grade any code that does not run correctly as a standalone file, and if the preceding command produces an error, the code portion of the assignment will receive a 0.

  To do the computational problems, we recommend that you install Clojure on your local machine and write and debug the answers to each problem in a local copy of `ps6-lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

Once you have entered your answers, please compile your copy of this LaTeX into a PDF and submit

(i) the compiled PDF renamed to `ps6-lastname-firstname.pdf`
(ii) the raw LaTeX file renamed to `ps6-lastname-firstname.tex` and
(iii) your `ps6-lastname-firstname.clj`

to the Problem Set 6 folder under 'Assignments' on MyCourses.

Example Problem: This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Example Answer: Put your answer in the box provided, like this:

Example answer is $L = \sum_0^\infty \mathcal{G}\delta_x$.

Background: Throughout this problem set we will be working with the Hidden Markov Model. In an HMM, there is a sequence of hidden states $c^{(1)}, ..., c^{(n)}$, and a sequence of observed words, $w^{(1)}, ..., w^{(n)}$. The set of all possible hidden states is S, and the set of all possible words is O. Thus $c^{(i)} \in S$ and $w^{(i)} \in O$ for all i. We defined the distribution on hidden states and words as follows in class:

$$\Pr(W^{(1)}, \ldots, W^{(k)}, C^{(1)}, \cdots, C^{(k)}) = \prod_{i=1}^{k} \Pr(W^{(i)} \mid C^{(i)} = c^{(i)}, \vec{\theta}_{O,c^{(i)}}) \Pr(C^{(i)} \mid C^{(i-1)} = c^{(i-1)}, \vec{\theta}_{T,c^{(i-1)}})$$

Where T are the state transitions, which here contain factors of S of length 1, in other words T is equivalent to S. Note that we are assuming (for the time being) that the parameters $\vec{\theta}_{T,c^{(i-1)}}$ and $\vec{\theta}_{O,c^{(i)}}$ are fixed, i.e. there is no uncertainty about their value. We will assume throughout that there are three hidden states (categories) and three words. These are defined by:

```
(def hidden-states '(Start N V))
(def vocabulary '(Call me Ishmael))
```

Note that, to make the exercises less painful, we are not including a `stop` state. There are three state transition distributions that we need to define, one for each of the hidden states: $\Pr(C^{(i)} \mid C^{(i-1)} = \rtimes)$, $\Pr(C^{(i)} \mid C^{(i-1)} = N)$, and $\Pr(C^{(i)} \mid C^{(i-1)} = V)$. As stated in the definition of the HMM probabilities above, these distributions are specified by the parameters $\vec{\theta}_{T,\rtimes}$, $\vec{\theta}_{T,N}$, and $\vec{\theta}_{T,V}$:

```
(def theta-transition-Start '(0 0.9 0.1))
(def theta-transition-N '(0 0.3 0.7))
(def theta-transition-V '(0 0.8 0.2))

(def theta-transition-dists-1 (list theta-transition-Start
                                    theta-transition-N
                                    theta-transition-V))
```

This means that the hidden state N has probability 0.3 of transitioning to hidden state N, and probability 0.7 of transitioning to V. Note that none of the states ever transition to the $\rtimes$ hidden state. Note also that throughout this problem set we will assume that the initial hidden category $C^{(0)} = \rtimes$ always. This means that the $\rtimes$ state only occurs as the first hidden state in a sequence, and never afterwards.

We also need to specify the observation distribution for the hidden states N and V (we assume the hidden state `Start` does produce observations). They are defined by:

```
(def theta-observation-N '(0.1 0.5 0.4))
(def theta-observation-V '(0.8 0.1 0.1))

(def theta-observation-dists-1 (list theta-observation-N
                                     theta-observation-V))
```

This means, for example, that the hidden state N has probability 0.4 of emitting the word *Ishmael*. The following helper functions will be useful for writing your code.

```
(defn dist-lookup [state states dists]
  (if (= state (first states))
      (first dists)
      (dist-lookup state (rest states) (rest dists))))

(defn log2 [n] (/ (Math/log n) (Math/log 2)))
```

```
(defn logsumexp [log-vals]
  (let [mx (apply max log-vals)]
    (+ mx
       (log2
           (apply +
                  (map (fn [z] (Math/pow 2 z))
                       (map (fn [x] (- x mx)) log-vals)))))))

(defn logscore-categorical [outcome outcomes params]
  (if (= outcome (first outcomes))
    (log2 (first params))
    (logscore-categorical outcome (rest outcomes) (rest params))))
```

Problem 1: Suppose that at time t the hidden state is $c^{(t)}$. Then we can compute the probability that the next hidden state is $c^{(t+1)}$ and the next observed word is $w^{(t+1)}$. This probability is equal to:

$$\begin{aligned}&\Pr(W^{(t+1)} = w^{(t+1)}, C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)})\\ =&\Pr(W^{(t+1)} = w^{(t+1)} \mid C^{(t+1)} = c^{(t+1)})\Pr(C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)})\end{aligned}$$

Write a procedure score-next-state-word which computes the log of this probability. It should take five arguments: the current hidden state, the next hidden state, and next observed word, the hidden state transition distributions, and the observation distributions. A call to `(score-next-state-word current-hidden next-hidden next-observed theta-transition-dists theta-observation-dists)` should return the log probability of the next hidden state and next observed word, given the current hidden state.

Hint: use the functions `dist-lookup` and `logscore-categorical` defined above.

Answer 1: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 2: Suppose that at time t the hidden state is $c^{(t)}$. Then we can compute the *marginal* probability of the word $w^{(t+1)}$ at time $t+1$ given the hidden state at time t. This probability is equal to:

$$\begin{aligned}&\Pr(W^{(t+1)} = w^{(t+1)} \mid C^{(t)} = c^{(t)})\\ =&\sum_{c^{(t+1)}} \Pr(W^{(t+1)} = w^{(t+1)}, C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)})\\ =&\sum_{c^{(t+1)}} \Pr(W^{(t+1)} = w^{(t+1)} \mid C^{(t+1)} = c^{(t+1)})\Pr(C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)})\end{aligned}$$

That is, we sum over all of the possible hidden states that could appear at time $t+1$.

Using the procedure `score-next-state-word` that you wrote in Problem 1, write a function compute-next-observation-marginal, which takes four arguments: `current-state`, `next-observation`, `theta-transition-dists`, and `theta-observation-dists`. When `(compute-next-observation-marginal current-state next-observation theta-transition-dists theta-observation-dists)` is called, it should return the marginal probability of the next observed word given the current hidden state.

Hint: use `logsumexp`.

Answer 2: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 3: Suppose that at time t the hidden state is $c^{(t)}$. Then we can compute the probability of the next k hidden states and next k observed words given the hidden state at time t. This probability is equal to:

$$\begin{aligned}
&\Pr(W^{(t+1)}, \dots, W^{(t+k)}, C^{(t+1)}, \cdots, C^{(t+k)} \mid C^{(t)} = c^{(t)}) \\
=& \prod_{i=t}^{t+k-1} \Pr(W^{(i+1)} = w^{(i+1)} \mid C^{(i+1)} = c^{(i+1)}) \Pr(C^{(i+1)} = c^{(i+1)} \mid C^{(i)} = c^{(i)}) \\
=& \Pr(C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)}) \Pr(W^{(t+1)} = w^{(t+1)} \mid C^{(t+1)} = c^{(t+1)}) \\
&\cdot \Pr(W^{(t+2)}, \dots, W^{(t+k)}, C^{(t+2)}, \cdots, C^{(t+k)} \mid C^{(t+1)} = c^{(t+1)})
\end{aligned}$$

Note that the equation shows that there is a recursive formula for this probability.

Write a procedure `score-next-states-words` which computes this probability. The function should take five arguments: the current hidden state, a list of k hidden states, a list of k observed words, the transition distributions, and the observation distributions. When `(score-next-states-words current-hidden next-hidden-states next-words theta-transition-dists theta-observation-dists)` is called, it should return the log probability that the k hidden states and k observed words will appear immediately following the current hidden state.

You should use recursion to write the function.

Answer 3: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 4: Suppose the at time t the hidden state is $c^{(t)}$. Then we can compute the marginal probability of the next k observed words given this hidden state, i.e. summing out all of the possible settings of the hidden states. This probability is equal to:

$$\begin{aligned}
&\Pr(W^{(t+1)}, \dots, W^{(t+k)} \mid C^{(t)} = c^{(t)}) \\
=& \sum_{c^{(t+1)}, \cdots c^{(t+k)}} \Pr(W^{(t+1)}, \dots, W^{(t+k)}, C^{(t+1)}, \cdots, C^{(t+k)} \mid C^{(t)} = c^{(t)}) \\
=& \sum_{c^{(t+1)}, \cdots c^{(t+k)}} \prod_{i=t}^{t+k-1} \Pr(W^{(i+1)} = w^{(i+1)} \mid C^{(i+1)} = c^{(i+1)}) \Pr(C^{(i+1)} = c^{(i+1)} \mid C^{(i)} = c^{(i)}) \\
=& \sum_{c^{(t+1)}} \Big[\Pr(C^{(t+1)} = c^{(t+1)} \mid C^{(t)} = c^{(t)}) \Pr(W^{(t+1)} = w^{(t+1)} \mid C^{(t+1)} = c^{(t+1)}) \\
&\quad \cdot \Pr(W^{(t+2)}, \dots, W^{(t+k)}) \mid C^{(t+1)} = c^{(t+1)}) \Big]
\end{aligned}$$

The final equation is very important. It shows that the marginal probability of the words following hidden state $c^{(t)}$ can be computed in a recursive manner. Write a function `compute-next-words-marginal`, which computes the log marginal probability of a sequence of observed words given the current hidden state. The function should take four arguments: the current hidden state, a list of words, the transition distributions, and the observation distributions. When `(compute-next-words-marginal current-hidden next-words theta-transition-dists theta-observation-dists)` is called, it should return the log of the marginal probability of the list of words given the hidden state, as defined in the equation above.

You should use recursion to write your function.

Answer 4: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 5: Find a sequence of two words $w^{(1)}, w^{(2)}$ such that the marginal probability of this sequence is different than the marginal probability of its reverse. That is, find $w^{(1)}, w^{(2)}$ such that:

$$\Pr(W^{(1)} = w^{(1)}, W^{(2)} = w^{(2)} \mid C^{(0)} = \rtimes) \neq \Pr(W^{(1)} = w^{(2)}, W^{(2)} = w^{(1)} \mid C^{(0)} = \rtimes)$$

Use your solution to Problem 4 to confirm that your sequences have different probabilities, and explain why this is true.

Answer 5: Please put your answer in the box below.

Problem 6: In the next several problems, we will be using your solution to Problem 4 to perform Bayesian inference in the HMM. Our goal will be to compute the posterior distribution over hidden states given a sequence of words. Recall that Bayes' Rule tells us how to calculate this posterior distribution:

$$\begin{aligned}&\Pr(C^{(1)} = c^{(1)}, \cdots, C^{(t)} = c^{(t)} \mid W^{(1)} = w^{(1)}, \dots, W^{(t)} = w^{(t)})\\ =&\frac{\Pr(C^{(1)} = c^{(1)}, \cdots, C^{(t)} = c^{(t)}) \Pr(W^{(1)} = w^{(1)}, \dots, W^{(t)} = w^{(t)} \mid C^{(1)} = c^{(1)}, \cdots, C^{(t)} = c^{(t)})}{\Pr(W^{(1)} = w^{(1)}, \dots, W^{(t)} = w^{(t)})}\end{aligned}$$

Write a procedure `compute-hidden-prior`, which takes two arguments: a list of k hidden states and the hidden state transition distributions. When `(compute-hidden-prior hidden-states theta-transition-dists)` is called, it should return the log prior probability of the hidden state sequence, i.e. the log value of

$$\Pr(C^{(1)} = c^{(1)}, \cdots, C^{(k)} = c^{(k)})$$

.

Answer 6: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 7: In this problem, we will continue with the Bayesian calculation from Problem 6. Write a procedure `compute-likelihood-of-words`, which takes three arguments: a list of k hidden states, a list of k words, and the observation distributions. When `(compute-likelihood-of-words hidden-states words theta-observation-dists)` is called, it should return the log probability of the k words being generated from the hidden states, i.e. the log value of $\Pr(W^{(1)} = w^{(1)}, \dots, W^{(k)} = w^{(k)} \mid C^{(1)} = c^{(1)}, \cdots, C^{(k)} = c^{(k)})$.

Answer 7: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 8: We are finally ready to solve our Bayesian inference problem. Write a procedure `compute-hidden-posterior`, which takes four arguments: a list of k hidden states, a list of k words, the hidden state transition distributions, and the observation distributions. When `(compute-hidden-posterior hidden-states words theta-transition-dists theta-observation-dists)` is called, it should return the log posterior probability of the k hidden states given the k words, i.e. it should return the log value of $\Pr(C^{(1)} = c^{(1)}, \cdots, C^{(k)} = c^{(k)} \mid W^{(1)} = w^{(1)}, \dots, W^{(k)} = w^{(k)})$.

Hint: Use your solutions to Problems 4, 6, and 7, and don't foget that we are in the log domain. Note also that in this function you will need to hardcode the first state as `'Start`.

Answer 8: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 9: Suppose that we use the function `compute-next-words-marginal` that you wrote in Problem 4, in order to compute the marginal probability of a sequence of k observed words. How many times is `compute-next-words-marginal` called during the execution of the program? What does this mean about the run-time of the program?

Hint: your formula should involve the number of hidden states and the length k of the sequence of words.

Answer 9: Please put your answer in the box below.

Problem 10: In class, we introduced *memoization* as a method for sharing computation and/or reducing the run-time of programs. Memoization refers to a technique where we intercept calls to a function with some arguments, check if the function has been called with those arguments before and, if not, call the function on those arguments, get the result, store it in a table and return it. On subsequent calls to that function with those arguments, we simply look up the result in the table and avoid recomputing the value. In Clojure, functions can be memoized by a call to `memoize`.

Using `memoize`, define a memoized version of your function `compute-next-words-marginal`. Name this function `compute-next-words-marginal-mem`.

Answer 10: Please put the answer in `ps6-lastname-firstname.clj`.

Problem 11: Suppose that we use the memoized function `compute-next-words-marginal-mem` that you wrote in the previous problem, in order to compute the marginal probability of a sequence of k observed words. How many times is `compute-next-words-marginal-mem` called during the execution of the program? How does this differ from the non-memoized version of the procedure?

Answer 11: Please put your answer in the box below.

LONG FORM READING QUESTION:

(This section is optional for students in LING/COMP 445, but must be completed if taking LING 645.)

You must answer this question on your own.

Kim et al.(2019) combines the traditional and neural network modelling approaches we have discussed in class. On the one hand their models make clear assumptions about the structure of language by defining neural network models that can only learn probabilistic context-free grammars, and on the other hand their models use distributed representations for both the observed and hidden variables, allowing models the freedom to encode additional information about the relations that may exist between these variables. This type of approach is sometimes called neuro-symbolic modelling. Based on all of the readings and discussions we have done and had this semester, in your opinion, (1) how might the traditional approach be better than the neural network approach? (2) how might the neural network approach be better than the traditional approach? and (3) how might a combined approach like Kim et al. be beneficial? There is no right or wrong answer, just make sure to properly motivate your position for each of these questions.

Answer: Please put your answer in the box below.