COMP/LING 445/645
Problem Set 3

There are several types of questions below. For programming questions, please put your answers into a file called `lastname-firstname.clj`. Be careful to follow the instructions exactly and be sure that all of your function definitions use the exact names, number of inputs, input types, number of outputs, and output types as requested in each question. Please make sure all other variables are also named as indicated.

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 `lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

For the code portion of the assignment, it is crucial to submit a standalone file that runs. Before you submit `lastname-firstname.clj` to us via email, make sure that your code executes correctly without any errors when run at the command line by typing `clojure 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 like produces an error, the code portion of the assignment will receive a 0.

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/`.

Once you have answered the question, please compile your copy of this LaTeX document into a PDF and submit (i) the compiled PDF (ii) the raw LaTeX file and (iii) your `lastname-firstname.clj` file via email to *both* timothy.odonnell@mcgill.ca and savanna.willerton@mail.mcgill.ca. The problem is set is due before 16:05 on Monday, November 11, 2019.

Problem 0: This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Answer 0: Put your answer right under the question like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Problem 1: In this problem set, we are going to be considering a variant on the hierarchical bag-of-words model that we looked at in class. In class, we used a Dirichlet distribution to define a prior distribution over θ, the parameters of the bag of words model. The Dirichlet distribution is a continuous distribution on the simplex—it assigns probability mass to all the uncountably many points on the simplex.

For this problem set, we will be looking at a considerably simpler prior distribution over the parameters θ. Our distribution will be *discrete*, and in particular will only assign positive probability to a finite number of values of θ. The probability distribution is defined in the code below:

```
(def vocabulary '(call me ishmael))

(def theta1 (list (/ 1 2 ) (/ 1 4 ) (/ 1 4 )))
(def theta2 (list (/ 1 4 ) (/ 1 2 ) (/ 1 4 )))

(def thetas (list theta1 theta2))

(def theta-prior (list (/ 1 2) (/ 1 2)))
```

Our vocabulary in this case consists of three words. Each value of θ therefore defines a bag of words distribution over sentences containing these three words. The first value of θ (`theta1`) assigns $\frac{1}{2}$ probability to the word 'call, $\frac{1}{4}$ to 'me, and $\frac{1}{4}$ to 'ishmael. The second value of θ (`theta2`) assigns $\frac{1}{2}$ probability to 'me, and $\frac{1}{4}$ to each of the other two words. The two values of θ each have prior probability of $\frac{1}{2}$. *Assume throughout the problem set that the vocabulary and possible values of θ are fixed to these values above.*

In addition to the code defining the prior distribution over θ, we will be using some helper functions defined in class:

```
(defn score-categorical [outcome outcomes params]
  (if (empty? params)
      (error "no matching outcome")
      (if (= outcome (first outcomes))
          (first params)
          (score-categorical outcome (rest outcomes) (rest params)))))


(defn list-foldr [f base lst]
  (if (empty? lst)
      base
      (f (first lst)
         (list-foldr f base (rest lst)))))

(defn score-BOW-sentence [sen probabilities]
  (list-foldr
   (fn [word rest-score]
     (+ (Math/log2 (score-categorical word vocabulary probabilities))
        rest-score))
   0
   sen))


(defn score-corpus [corpus probabilities]
  (list-foldr
   (fn [sen rst]
     (+ (score-BOW-sentence sen probabilities) rst))
   0
   corpus))

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

Recall that the function `score-corpus` is used to compute the log probability of a corpus given a particular value of the parameters θ. Also recall (from the Discrete Random Variables module) the purpose of the function `logsumexp`, which is used to compute the sum of (log) probabilities; you should return to the lecture notes if you don't remember what this function is doing. (Note that the version of `logsumexp` here differs slightly from the lecture notes, as it does not use the & notation.)

Our initial corpus will consist of two sentences:

```
(def my-corpus '((call me)
                 (call ishmael)))
```

Write a function `theta-corpus-joint`, which takes three arguments: `theta`, `corpus`, and `theta-probs`. The argument `theta` is a value of the model parameters θ, and the argument `corpus` is a list of sentences. The argument `theta-probs` is a prior probability distribution over the values of θ. The function should return the **log** of the joint probability $\Pr(C = corpus, \theta = theta)$.

Use the chain-rule identity discussed in class: $\Pr(C, \theta) = \Pr(C|\theta)\Pr(\theta)$. Assume that the prior distribution $\Pr(\theta)$ is defined by the probabilities in `theta-probs`.

After defining this function, call `(theta-corpus-joint theta1 my-corpus theta-prior)`. This will compute (the log of) the joint probability of the model parameters `theta1` and the corpus `my-corpus`.

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

Problem 2: Write a function `compute-marginal`, which takes two arguments: `corpus` and `theta-probs`. The argument `corpus` is a list of sentences, and the argument `theta-probs` is a prior probability distribution on values of θ. The function should return the **log** of the marginal likelihood of the corpus, when the prior distribution on θ is given by theta-probs. That is, the function should return $\log[\sum_{\theta \in \Theta} \Pr(\mathbf{C} = \mathsf{corpus}, \Theta = \theta)]$.

Hint: Use the `logsumexp` function defined above.

After defining `compute-marginal`, call `(compute-marginal my-corpus theta-prior)`. This will compute the marginal likelihood of my-corpus (which was defined above), given the prior distribution `theta-prior`.

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

Problem 3: Write a procedure `compute-conditional-prob`, which takes three arguments: `theta`, `corpus`, and `theta-probs`. The arguments have the same interpretation as in Problems 1 and 2. The function should return the **log** of the conditional probability of the parameter value theta, given the corpus. Remember that the conditional probability is defined by the equation:

$$\Pr(\Theta = \theta | \mathbf{C} = \mathsf{corpus}) = \frac{\Pr(\mathbf{C} = \mathsf{corpus}, \Theta = \theta)}{\sum_{\theta \in \Theta} \Pr(\mathbf{C} = \mathsf{corpus}, \Theta = \theta)} \tag{1}$$

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

Problem 4: Write a function `compute-conditional-dist`, which takes two arguments: `corpus` and `theta-probs`. For every value of θ in `thetas` (i.e., `theta1` and `theta2`), it should return the conditional probability of θ given the corpus. That is, it should return a list of conditional probabilities of the different values of θ.

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

Problem 5: Call `(compute-conditional-dist my-corpus theta-prior)`. What do you notice about the conditional distribution over values of θ? Exponentiate the values you get back, so that you can see the regular probabilities, rather than just the log probabilities. Explain why the conditional distribution looks the way it does, with reference to the properties of `my-corpus`.

Answer 5: The values of the conditional distribution over values of θ are much smaller in magnitude than the marginal and joint probabilities. After exponentiating, we can clearly see that the conditional distribution sums to 1. This is because conditional distribution by definition normalizes the joint probabilities over the sum of marginalized joint probabilities. After exponentiating, theta1 had the conditional probability of approximately 2/3 and theta2 had the conditional probability of 1/3. In the context of our corpus, this makes sense since call appears twice, where as me and Ishmael occur only once. Theta1 gives the respective values (1/2 1/4 1/4) to vocabulary words (call me Ishmael), and since they appear in the corpus in this exact proportion, the distribution of theta1 is more likely than theta2.

Problem 6: When you call `compute-conditional-dist`, you get back a probability distribution over values of θ (the conditional distribution over θ given an observed corpus). This is a probability distribution just like any other. In particular, it can be used as the prior distribution over values of θ in a hierarchical bag of words model. Given this new hierarchical BOW model, we can do all of the things that we normally do with such a model. In particular, we can compute the marginal likelihood of a corpus under this model. This marginal likelihood is called a *posterior predictive distribution*.

Below we have defined the skeleton of a function `compute-posterior-predictive`. It takes three arguments: `observed-corpus`, `new-corpus`, and `theta-probs`. `observed-corpus` is a corpus which we observe, and use to compute a conditional distribution over values of θ. Given this conditional distribution over θ, we will then compute the marginal likelihood of the corpus `new-corpus`. The procedure `compute-posterior-predictive` should return the marginal likelihood of new-corpus given the conditional distribution on θ.

```
(defn compute-posterior-predictive [observed-corpus new-corpus theta-probs]
  (let [conditional-dist ...
   (compute-marginal ...
```

Once you have implemented `compute-posterior-predictive`, call `(compute-posterior-predictive my-corpus my-corpus theta-prior)`. What does this quantity represent? How does its value compare to the marginal likelihood that you computed in Problem 2?

Answer 6: The posterior predictive distribution represents the marginal likelihood of a corpus over the possible values of θ in a hierarchical BOW model. The likelihood is slightly higher than the marginal likelihood, reflecting the difference between the conditional distribution used in the posterior predictive distribution formula vs the prior probability distribution of the thetas. The conditional distribution more accurately reflects the observed corpus in the theta probabilities, whereas the prior distribution does not.

Problem 7: In the previous problems, we have written code that will compute marginal and conditional distributions *exactly*, by enumerating over all possible values of θ. In the next problems, we will develop an alternate approach to computing these distributions. Instead of computing these distributions exactly, we will approximate them using random sampling.

The following functions were defined in class, and will be useful for us going forward:

```
(defn normalize [params]
  (let [sum (apply + params)]
   (map (fn [x] (/ x sum)) params)))

(defn flip [weight]
  (if (< (rand 1) weight)
     true
     false))

(defn sample-categorical [outcomes params]
```

```
  (if (flip (first params))
      (first outcomes)
      (sample-categorical (rest outcomes)
                          (normalize (rest params)))))

(defn repeat [f n]
  (if (= n 0)
      '()
      (cons (f) (repeat f (- n 1)))))

(defn sample-BOW-sentence [len probabilities]
        (if (= len 0)
          '()
          (cons (sample-categorical vocabulary probabilities)
            (sample-BOW-sentence (- len 1) probabilities))))
```

Recall that the function `sample-BOW-sentence` `samples` a sentence from the bag of words model of length len, given the parameters theta.

Define a function `sample-BOW-corpus`, which takes three arguments: `theta`, `sent-len`, and `corpus-len`. The argument `theta` is a value of the model parameters θ. The arguments `sent-len` and `corpus-len` are positive integers. The function should return a sample corpus from the bag of words model, given the model parameters `theta`. Each sentence should be of length `sent-len` and number of sentences in the corpus should be equal to `corpus-len`. For example, if `sent-len` equals 2 and `corpus-len` equals 2, then this function should return a list of 2 sentences, each consisting of 2 words.

Hint: Use `sample-BOW-sentence` and `repeat`.

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

Problem 8: Below we have defined the skeleton of the function `sample-theta-corpus`. This function takes three arguments: `sent-len` `corpus-len` and `theta-probs`. It returns a list with two elements: a value of θ sampled from the distribution defined by `theta-probs`; and a corpus sampled from the bag of words model given the sampled θ. (The number of sentences in the corpus should equal `corpus-len`, and each sentence should have `sent-len` words in it.)

We will call the return value of this function a `theta-corpus` pair.

```
(defn sample-theta-corpus [sent-len corpus-len theta-probs]
  (let [theta ...
    (list theta ...
```

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

Problem 9: Below we have defined some useful functions for us. The function `get-theta` takes a `theta-corpus` pair, and returns the value of `theta` in it. The function `get-corpus` takes a `theta-corpus` pair, and returns its corpus value. The function `sample-thetas-corpora` samples multiple theta-corpus pairs, and returns a list of them. In particular, the number of samples it returns equals sample-size.

```
(defn get-theta [theta-corpus]
  (first theta-corpus))
```

```
(defn get-corpus [theta-corpus]
  (first (rest theta-corpus)))

(defn sample-thetas-corpora [sample-size sent-len corpus-len theta-probs]
  (repeat (fn [] (sample-theta-corpus sent-len corpus-len theta-probs)) sample-size))
```

We are now going to estimate the marginal likelihood of a corpus by using random sampling. Here is the general approach that we are going to use. We are going to sample some number (for example 1000) of `theta-corpus` pairs. These are 1000 samples from the joint distribution defined by the hierarchical bag of words model. We are then going to throw away the values of `theta` that we sampled; this will leave us with 1000 corpora sampled from our model.

We are going to use these 1000 sampled corpora to estimate the probability of a specific target corpus. The process here is simple. We just count the number of times that our target corpus appears in the 1000 sampled corpora. The ratio of the occurrences of the target corpus to the number of total corpora gives us an estimate of the target's probability.

More formally, let us suppose that we are given a target corpus $\mathbf{t}$. We will define the indicator function $\mathbb{1}_{\mathbf{t}}$ by:

$$\mathbb{1}_{\mathbf{t}}(c) = \begin{cases} 1, & \text{if } t = c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We will sample n corpora $c_1, ..., c_n$ from the hierarchical bag of words model. We will estimate the marginal likelihood of the target corpus $\mathbf{t}$ by the following formula:

$$\sum_{\theta \in \Theta} \Pr(\mathbf{C} = \mathbf{t}, \Theta = \theta) \approx \frac{1}{n} \sum_{i}^{n} \mathbb{1}_{\mathbf{t}}(c_i) \tag{3}$$

Define a procedure `estimate-corpus-marginal`, which takes five arguments: `corpus`, `sample-size`, `sent-len`, `corpus-len`, and `theta-probs`. The argument `corpus` is the target corpus whose marginal likelihood we want to estimate. `sample-size` is the number of corpora that we are going to sample from the hierarchical model (its value was 1000 in the discussion above). The arguments `corpus-len` and `sent-len` characterize the number of sentences in the corpus and the number of words in each sentence, respectively. The argument `theta-probs` is the prior probability distribution over θ for our hierarchical model.

The procedure should return an estimate of the marginal likelihood of the target corpus, using the formula defined in Equation 3.

Hint: Use `sample-thetas-corpora` to get a list of samples of `theta-corpus pairs`, and then use `get-corpus` to extract the corpus values from these pairs (and ignore the `theta` values).

Answer 9: Please put your answer in `lastname-firstname.clj`.

Problem 10: Call `(estimate-corpus-marginal my-corpus 50 2 2 theta-prior)` a number of times. What do you notice? Now call `(estimate-corpus-marginal my-corpus 10000 2 2 theta-prior)` a number of times. How do these results compare to the previous ones? How do these results compare to the exact marginal likelihood that you computed in Problem 2?

Answer 10: Calling estimate-corpus-marginal with a sample size of 50 often returns 0 or 1/50. This is because the sample size is too small to consistently sample corpora that are the same as the target. Calling with a sample size of 10000 yields more accurate estimates that are within +/- 0.002 of the exact marginal likelihood that we calculated earlier, a very striking improvement. This is because a greater sample size allows more a chance for the corpus in question to appear in the samples before determining the estimate.

Problem 11: These functions will be useful for us in the next problem.

```
(defn get-count [obs observation-list count]
  (if (empty? observation-list)
      count
      (if (= obs (first observation-list))
         (get-count obs (rest observation-list) (+ 1 count))
         (get-count obs (rest observation-list) count))))


(defn get-counts [outcomes observation-list]
  (let [count-obs (fn [obs] (get-count obs observation-list 0))]
    (map count-obs outcomes)))
```

In Problem 9, we introduced a way of approximating the marginal likelihood of a corpus by using random sampling. We can similarly approximate a conditional probability distribution by using random sampling.

Suppose that we have observed a corpus **c**, and we want to compute the conditional probability of a particular θ. We can approximate this conditional probability as follows. We first sample n theta-corpus pairs. We then remove all of the pairs in which the corpus does not match our observed corpus **c**. We finally count the number of times that θ occurs in the remaining theta-corpus pairs, and divide by the total number of remaining pairs. This process is called *rejection sampling*.

Define a function `rejection-sampler` which has the following form:

`(rejection-sampler theta observed-corpus sample-size sent-len corpus-len theta-probs)`

We want to get an estimate of the conditional probability of `theta`, given that we have observed the corpus `observed-corpus`. `sample-size` is a positive integer, and we will estimate this conditional probability by taking `sample-size` samples from the joint distribution on `theta-corpus` pairs. The procedure should filter out any `theta-corpus` pairs in which the corpus does not equal the observed corpus. In the remaining pairs, it should then count the number of times that theta occurs, and divide by the total number of remaining pairs.
Hint: Use `get-counts` to count the number of occurrences of `theta`.

Answer 11: Please put your answer to the coding problem `lastname-firstname.clj`.

Problem 12: Call `(rejection-sampler theta1 my-corpus 100 2 2 theta-prior)` a number of times. What do you notice? How large does `sample-size` need to be until you get a stable estimate of the conditional probability of `theta1`? Why does it take so many samples to get a stable estimate?

Answer 12: When calling the rejection sampler at sample size 100, the result is often 0 or 1/100. This is most likely because the sample size is not large enough to yield many matches in light of the marginal likelihood of the corpus that is about 1/100. In order to get a stable estimate of the conditional probability, the sample size would need to be at least 10,000 so that the target corpus can appear more often in the sample to give a more accurate representation of the probability of the theta.