MITx: 6.008.1x Computational Probability and Inference

Mini-project: Movie Recommendations

You decide to reward yourself by watching a movie. But which one? This is a question that clearly merits a full scientific investigation. You decide to perform Bayesian inference to determine which movie you should watch next.

Let $\boldsymbol{X} \in \{0, \dots, M-1\}$ be a discrete random variable that represents the true rating (i.e. quality) of a movie you are considering, where 0 means awful, and $M-1$ means amazing(you could think of these as the number of stars). $\boldsymbol{X}$ is distributed according to $p_{\boldsymbol{X}}(\cdot)$.

A number of helpful strangers have watched several movies and recorded their opinion of the movies' ratings. Let $\boldsymbol{Y}_n \in \{0, \dots, M-1\}$ be a discrete random variable that represents the rating user n provided for this movie, where $n = 0, \dots, N-1$. User ratings are noisy. We will assume that conditioned on the movie's true/inherent rating X, the ratings of different users are independent and identically distributed such that $p_{\boldsymbol{Y}_n|\boldsymbol{X}}(y|x) = p_{\boldsymbol{Y}|\boldsymbol{X}}(y|x)$ for $n = 0, 1, \dots, N-1$. In other words, the joint probability distribution for $\boldsymbol{X}, \boldsymbol{Y}_0, \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_{N-1}$ has the factorization,

$$p_{\boldsymbol{X},\boldsymbol{Y}_0,\boldsymbol{Y}_1,\dots,\boldsymbol{Y}_{N-1}}(x, y_0, y_1, \dots, y_{N-1}) = p_X(x) \prod_{n=0}^{N-1} p_{Y|X}(y_n|x).$$

(a) Determine $p_{\boldsymbol{X}|\boldsymbol{Y}_0,\dots,\boldsymbol{Y}_{N-1}}(x|y_0, \dots, y_{N-1})$ in terms of the given functions $p_{\boldsymbol{X}}(\cdot)$ and $p_{\boldsymbol{Y}|\boldsymbol{X}}(\cdot|\cdot)$. In other words, given that you observed $\boldsymbol{Y}_0 = y_0, \dots, \boldsymbol{Y}_{N-1} = y_{N-1}$, what is the probability that $\boldsymbol{X} = x$? There is no coding for this part. Also there is nothing to submit for this part. Please show this though so that you know what you are implementing for this mini-project.

Code and Data: The remainder of this mini-project requires you to work off code and data we are providing. Download *movie-rating.zip* and extract it to your working directory. It contains:

- Folder `data` whose contents you need not look at.
- File `movie data_helper.py`, which you need not edit, but whose function you will need in your code:
  - `get_movie_id_list()` returns a 1D array containing ID numbers for the movies that you should process. The movie IDs are sequential numbers ranging from 0 to the number of movies included in the input minus 1.
  - `get_ratings(movie_id)` returns a 1D array containing the ratings given by users to the movie `movie_id`. The number of users who rated a given movie might vary, and as a result, the length of the returned vectors is different for each movie.
  - `get_movie_name(movie_id)` returns the title of the movie `movie_id`.
- File `movie_recommendations.py` that contains a number of incomplete functions that you will fill in. This is the only file you will edit. You will need to read the comments in this file to understand what code you have to add.

Important: Do not change the function definitions in the code as they must remain consistent for grading. The only parts of the code that you need to fill in are in blocks denoted by "YOUR CODE GOES HERE".

(b) Implement your answer to part **(a)** by filling out the function `compute_posterior` in `movie_recommendations.py`. See the comment at the beginning of the function for what the expected input and output are.

Note that for just this part, the code should allow for $\boldsymbol{Y}$ to take on a value in $\{0, 1, \ldots, K-1\}$ whereas $\boldsymbol{X}$ still takes on a value in $\{0, 1, \ldots, M-1\}$. This code implements a general Bayes' rule calculation with observations $\boldsymbol{Y}_0, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{N-1}$ that are conditionally independent and identically distributed given the hidden variable $\boldsymbol{X}$. After this part, we will let $K = M$.

Important: Your solution to part **(a)** should have a numerator and denominator that each has many, many probabilities being multiplied together. On a computer, numerical issues cause these multiplications by small numbers to at some point just produce 0 even when the product is not actually 0. This will cause problems! To avoid this issue, work in the log domain(natural log, implemented using NumPy's `np.log`). Recall that $\log(ab) = \log a + \log b$ for any $a > 0$, and $b > 0$ and so $\log(a/b) = \log a - \log b$. The bottom line is that your code should *not* be multiplying together a large list of probabilities. It should be adding the log of these probabilities instead.

Note that once you take the log of the denominator term in your answer to **(b)**, you should encounter a log of a sum of exponential terms. Please use SciPy function `scipy.misc.logsumexp` to compute this log of the denominator (`scipy.misc.logsumexp` provides a numerically stable way to compute the log of the sum of exponential terms; check the SciPy documentation for how to use this function).

You will now use the general pupose Bayes' rule calculator you just coded for the movie recommendations! We will assume a uniform prior $p_{\boldsymbol{X}}(x) = 1/M$, for $x = 0, \ldots, M-1$ to reflect our beliefs before we have seen any user ratings. Moreover, we will use the following likelihood for the ratings:

$$p_{\boldsymbol{Y}|\boldsymbol{X}}(y|x) = \begin{cases} \frac{\alpha}{|y-x|} & \text{if } y \neq x \\ 2\alpha & \text{if } y = x \end{cases}$$

where $\alpha > 0$ is a normalization constant.

(c) Fill in the function `compute_movie_rating_likelihood` that computes the likelihood (conditional probability $p_{\boldsymbol{Y}|\boldsymbol{X}}(y|x)$) given above and returns it as a 2D array. See the comment at the beginning of the function for what the expected input and output are.

(d) Determine the posterior distribution and the MAP estimate of the quality of each movie. To do this, complete function `infer_true_movie_ratings` and use it on the provided data. Note that this function takes in the number of observed ratings to use per movie (`num_observations`), which will make part **(g)** easier. For this part, use `num_observations = -1` to utilize all of the available ratings for each movie. See the comment at the beginning of the function for what the expected input and outputs are. Take advantage of the the function `compute_posterior` that you completed in part **(b)**. Note that you can process each movie separately, since we assume that both the quality and the ratings are independent across different movies.

(e) Provide the movie titles of the 10 movies with the highest MAP ratings.

Finally, we investigate how the posterior distribution of the movie quality changes as the number of ratings available for that movie grows. To do this, we look at entropy.

(f) First, complete the function `compute_entropy`. See the comment at the beginning of the function for what the expected input and output are.

Hint: Do not forget that $0 \log 0 \triangleq 0$. You can assume that `distribution[j]` $= 0$ if `distribution[j]` $\leq 10^{-8}$.

(g) Complete the function `compute_true_movie_rating_posterior_entropies` to compute the entropy of the posterior distribution $p_{\boldsymbol{X}|\boldsymbol{Y}_0,\ldots,\boldsymbol{Y}_{N-1}}(\,\cdot\,|\,y_0,\ldots,y_{N-1})$ for all movies, where N is the number of observations to use per movie. See the comment at the beginning of the function for what the expected input and output are.

- Plot the posterior entropy averaged over all movies as a function of N, for $1 \leq N \leq 200$. How does the entropy behave as we receive more and more observations? What does this mean in terms of the "randomness" of the movie qualit $\boldsymbol{X}$ s we get more and more observations?

  For plotting, use the `plt.plot` function provided by the `matplotlib.pyplot` package. We have already imported this package for you and named as `plt`.