



Bookmarks

► Introduction

▼ Part 1: Probability and Inference

**Week 1: Introduction to Probability**

Exercises due Sep 22, 2016 at 02:30 IST

**Week 1: Probability Spaces and Events**

Exercises due Sep 22, 2016 at 02:30 IST

**Week 1: Random Variables**

Exercises due Sep 22, 2016 at 02:30 IST

**Week 2: Jointly Distributed Random Variables**

Exercises due Sep 29, 2016 at 02:30 IST

**Week 2: Conditioning on Events**

Exercises due Sep 29, 2016 at 02:30 IST



Part 1: Probability and Inference &gt; Weeks 3 and 4: Mini-project on Movie Recommendations &gt; Mini-project 1

## Mini-project 1

🔖 Bookmark this page

### Mini-project: Movie Recommendations








100/100 points (graded)

**The submission system is up (October 17, 2016):** The autograder is up and runs Python 3, NumPy, and SciPy. Unfortunately it does not support other packages in Anaconda. If you are using other packages, please edit your code to only use NumPy and SciPy. Also, our autograder uses a random subset of the data we gave you, so if you hard-coded any constants based on the data we gave you, then you might run into problems. Also, do *not* change the name of the imported package *movie\_data\_helper* (leave the import line intact and do not import it as having some other name). Scroll down to the bottom for the big text box where you paste your code. **Note that we are not grading parts (a), (c), (e), and (h).**

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  $X \in \{0, \dots, M - 1\}$  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).  $X$  is distributed according to  $p_X(\cdot)$ .

A number of helpful strangers have watched several movies and recorded their opinion of the movies' ratings. Let  $Y_n \in \{0, \dots, M - 1\}$  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

**Week 2: Homework 1**Homework due Sep 29, 2016 at 02:30 IST **Week 3: Inference with Bayes' Theorem for Random Variables**Exercises due Oct 06, 2016 at 02:30 IST **Week 3: Independence Structure**Exercises due Oct 06, 2016 at 02:30 IST **Week 3: Homework 2**Homework due Oct 06, 2016 at 02:30 IST **Notation Summary Up Through Week 3****Weeks 3 and 4: Mini-project on Movie****Recommendations**Mini-projects due Oct 21, 2016 at 02:30 IST **Week 4: Decisions and Expectations**Exercises due Oct 13, 2016 at 02:30 IST **Week 4: Measuring Randomness**Exercises due Oct 13, 2016 at 02:30 IST 

conditioned on the movie's true/inherent rating  $\mathbf{X}$ , the ratings of different users are independent and identically distributed such that  $p_{Y_n|\mathbf{X}}(\mathbf{y} | \mathbf{x}) = p_{Y|\mathbf{X}}(\mathbf{y} | \mathbf{x})$  for  $n = 0, 1, \dots, N-1$ . In other words, the joint probability distribution for  $\mathbf{X}, Y_0, Y_1, \dots, Y_{N-1}$  has the factorization

$$p_{\mathbf{X}, Y_0, Y_1, \dots, Y_{N-1}}(\mathbf{x}, y_0, y_1, \dots, y_{N-1}) = p_{\mathbf{X}}(\mathbf{x}) \prod_{n=0}^{N-1} p_{Y|\mathbf{X}}(y_n | \mathbf{x}).$$

- **(a)** Determine  $p_{\mathbf{X}|Y_0, \dots, Y_{N-1}}(\mathbf{x} | y_0, \dots, y_{N-1})$  in terms of the given functions  $p_{\mathbf{X}}(\cdot)$  and  $p_{Y|\mathbf{X}}(\cdot | \cdot)$ . In other words, given that you observed  $Y_0 = y_0, \dots, Y_{N-1} = y_{N-1}$ , what is the probability that  $\mathbf{X} = \mathbf{x}$ ? 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 functions 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 vector is different for each movie.
  - get\_movie\_name(movie\_id) returns the title of the movie movie\_id.

## Week 4: Towards Infinity in Modeling Uncertainty

Exercises due Oct 13, 2016 at 02:30 IST



## Week 4: Homework 3

Homework due Oct 13, 2016 at 02:30 IST



### ► Part 2: Inference in Graphical Models

- 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  $\mathbf{Y}$  to take on a value in  $\{0, 1, \dots, K - 1\}$  whereas  $\mathbf{X}$  still takes on a value in  $\{0, 1, \dots, M - 1\}$ . This code implements a general Bayes' rule calculation with observations  $\mathbf{Y}_0, \mathbf{Y}_1, \dots, \mathbf{Y}_{N-1}$  that are conditionally independent and identically distributed given the hidden variable  $\mathbf{X}$ . 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 the 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 purpose Bayes' rule calculator you just coded for movie recommendations! We will assume a uniform prior  $p_X(x) = 1/M$ , for  $x = 0, \dots, M - 1$  to reflect our beliefs before we have seen any user ratings. Moreover, we will use the following likelihood for the ratings:

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

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

- **(c)** Fill in the function `compute_movie_rating_likelihood` that computes the likelihood (conditional probability  $p_{Y|X}(y | x)$ ) 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.

**The remainder of the mini-project requires material from week 4.**

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] ≤ 10-8`**.

- **(g)** Complete the function `compute_true_movie_rating_posterior_entropies` to compute the entropy of the posterior distribution  $p_{\mathbf{X}|\mathbf{Y}_0, \dots, \mathbf{Y}_{N-1}}(\cdot | \mathbf{y}_0, \dots, \mathbf{y}_{N-1})$  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.
- **(h)** 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 movie quality  $\mathbf{X}$  as 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 it as `plt`.

**This is where you paste your code:**



```
1 # students please paste your movie_recommendations.py code here
2 #!/usr/bin/env python
3 """
```

Press ESC then TAB or click outside of the code editor to exit

Correct

## Test results

Hide output

**CORRECT**

Test: `compute_posterior(array([ 0.5, 0.5]), array([[ 0.45, 0.6 ], [ 0.55, 0.4 ]]), array([0, 0, 1, 1, 1, 0, 1, 1]))`

Output:

`[0.6746346188187191, 0.3253653811812806]`

Test: `compute_posterior(array([ 0.5, 0.5]), array([[ 0.45, 0.6 ], [ 0.55, 0.4 ]]), array([0, 0, 0, ..., 1, 1, 1]))`

Output:

`[0.910873103747323, 0.0891268962527537]`

Test: `infer_true_movie_ratings()`

**Output:**

```
[[[1.6350281795413772e-60, 9.174504896585845e-60, 1.7198727572217451e-53,
4.2514193456454204e-44, 1.4561913930989338e-30, 3.7761894328355045e-16,
0.09250564695345667, 0.9042979585396259, 0.003196394506912414,
1.690351931949031e-17, 2.914635705629123e-25], [1.244950206859825e-62,
3.6457941719199262e-62, 1.6283086631786014e-55, 7.147334168313788e-46,
3.9890318434000295e-33, 9.965927428036768e-20, 2.5748740559009804e-05,
0.00815538197909091, 0.9918188692803169, 2.66414926789489e-14,
2.7374520833924526e-23], [3.6937648475701687e-88, 1.066639964264882e-89,
6.408993298332822e-86, 3.449894684068869e-80, 6.701652900902232e-72,
7.920559368383788e-60, 5.279674218672355e-45, 2.6183408711218695e-31,
6.223263559708415e-15, 5.158013253932884e-09, 0.9999999948419713],
[2.3332018441812333e-75, 1.9108089256041187e-76, 5.17792822945208e-72,
3.3677379695238645e-65, 1.9119880937762953e-55, 1.9954251316077386e-42,
6.093618965595895e-27, 6.302940736068145e-16, 1.1892248245330326e-05,
3.802710193176987e-05, 0.9999500806498292], [2.573071580558816e-102,
2.1844948702733457e-104, 3.1394136997038553e-101, 1.1345091548139672e-95,
7.126613094113799e-88, 7.144610861970301e-77, 9.769359850541518e-63,
2.987308295751671e-49, 2.7326467420058906e-33, 2.5535062276159402e-17, 1.0],
[7.489032861282384e-58, 6.321757963499626e-57, 1.0832637247005175e-49,
2.4688528843343422e-39, 8.118035760668743e-26, 4.589946050067302e-17,
4.575208124659331e-06, 0.008684809842556146, 0.9913106149478331,
1.4887963888307024e-12, 5.6683684470326835e-21], [7.562298426184032e-52,
1.0109789497806224e-49, 3.1030983171820566e-41, 6.144867682732136e-32,
7.006262593083221e-19, 2.0535629686177645e-09, 0.9595167904786289,
0.04047524852783278, 7.958939987574732e-06, 8.506040250226318e-19,
1.910968626346684e-25], [3.4615105241140095e-75, 9.22932995959981e-76,
1.2528342123908752e-70, 9.012580742610116e-63, 1.3596123552305404e-51,
1.5579159067731038e-36, 8.395202040823686e-18, 2.6435990229314813e-07,
0.9999997356399669, 1.210397372570605e-13, 2.862254928842224e-21],
[4.016805318052122e-88, 8.386756722918406e-90, 4.2681253241138475e-86,
1.1287533255554161e-79, 1.3660853202614425e-70, 1.7423212848339604e-58,
2.4720543731806663e-44, 9.708502524646973e-32, 5.469661711451792e-16,
6.499092509654974e-07, 0.9999993500907587], [3.9043373065810146e-94,
1.2994031388557121e-95, 7.572859713823357e-92, 7.375318374533702e-87,
1.4120355156108802e-79, 1.0453934503031492e-68, 1.381368923805977e-54,
```



3.082770929933574e-41, 8.693798806182955e-25, 7.771560188250232e-12, 0.999999999922409], [5.849696444284227e-67, 3.250704761932138e-67, 1.056828572008144e-61, 7.051958076996359e-54, 7.78592354471346e-43, 8.780117758708988e-29, 2.347449360577803e-11, 8.449906205564015e-06, 0.9999915256078508, 2.444949770502014e-08, 1.2965474232964994e-11], [3.595935356650143e-69, 3.474060613947014e-69, 3.3157517713924105e-63, 7.598632027973043e-54, 2.222187830171395e-40, 6.89171012749651e-25, 6.345311637745549e-11, 1.9055373840849177e-06, 0.9999980943991569, 7.659533302136028e-17, 1.8568324281183057e-26], [7.323221177449631e-47, 1.8449677630768723e-43, 2.9463196413521747e-33, 8.405390798440811e-24, 1.3081911286929532e-12, 1.6455285202189117e-06, 0.9999968714801977, 1.482989686501149e-06, 2.8477632420705895e-13, 3.4460225297993705e-24, 1.9137240731280476e-29], [1.1089964183342524e-53, 9.720940515704276e-52, 1.0164898407552513e-43, 2.9647793968755574e-33, 1.52289120913383e-20, 4.761230609728388e-10, 0.9999681747127797, 3.1820593271781586e-05, 4.21782314308496e-09, 1.5910641952674778e-21, 5.812770059107361e-28], [7.92376056740058e-61, 3.4943766796309344e-60, 3.534515560482154e-54, 2.294461707815017e-46, 2.522177918415085e-35, 1.8288566991359437e-19, 0.5600208065161014, 0.4105903440092512, 0.029388849474583527, 5.70407122457642e-14, 1.3627181042869553e-19], [9.375072508657247e-77, 1.630704080235025e-77, 1.5027752475965076e-72, 5.258892990916577e-65, 2.8293316568135602e-55, 1.0954098762972636e-43, 6.4350971148288535e-28, 3.147861936012656e-12, 0.9999999678337878, 3.2162746971806955e-08, 3.25219422397339e-13], [1.4821102686742487e-74, 2.1203469256953575e-75, 1.7977536409904253e-70, 2.4130793039677663e-62, 1.933671406794735e-51, 1.2114357233428613e-36, 1.4575007323730602e-19, 3.824652356277252e-09, 0.9999999950315157, 1.143825813778081e-09, 6.945848594994193e-15], [2.2744405446807087e-70, 3.4678198747681243e-71, 2.530535906253258e-66, 1.178648479098009e-58, 3.1781668373282096e-48, 1.109556347913227e-36, 4.391311251588824e-22, 1.7742801122592062e-13, 0.030129049901890546, 0.0034643925249165444, 0.9664065575730197], [1.7511530830551355e-58, 5.847774085763437e-58, 6.643770531505924e-52, 2.900307208774445e-43, 4.719825533336444e-32, 1.9491598745277613e-18, 0.0007027276071919846, 0.03256690794513215, 0.9667303632356365, 1.2118205910820106e-09, 2.1487240859461994e-13], [1.9720650242067765e-68, 2.2735609879717244e-69,

```
1.371578924262951e-64, 2.236453754668868e-56, 4.101433746088556e-45,  
1.6108714839798382e-31, 7.427850524445796e-15, 2.4728908568676923e-08,  
0.972040710338344, 0.0008943661958219692, 0.027064898736928553]], [7.0, 8.0,  
10.0, 10.0, 10.0, 8.0, 6.0, 8.0, 10.0, 10.0, 8.0, 8.0, 6.0, 6.0, 6.0, 8.0,  
8.0, 10.0, 8.0, 8.0]]
```

Test: `compute_entropy(array([ 0.1, 0.2, 0.3, 0.4, 0. , 0. , 0. ]))`

**Output:**

```
1.8464393446710154
```

Test: `compute_true_movie_rating_posterior_entropies(3)`

**Output:**

```
[2.061654759387925, 2.061654759387925, 0.46825642687776736,  
2.5130665676815083, 1.922818514230999, 2.3511953779784114,  
2.061654759387925, 2.061654759387925, 1.7396678041209848,  
2.1441413580251862, 3.00487830502065, 2.061654759387925, 3.125354153825567,  
2.1441413580251862, 2.5002308940436357, 1.7396678041209848,  
2.1441413580251862, 1.922818514230999, 2.10990093343172, 2.1441413580251862]
```

Test: `compute_true_movie_rating_posterior_entropies(6)`

**Output:**

```
[1.8997493444435014, 1.0836210546074154, 0.4499760257009765,  
2.0382721878835346, 0.5232336178792423, 2.0225087796992662,  
1.8839568220213303, 2.038272187883535, 1.6549040635939916,  
1.6507985269018932, 2.6437304831767268, 0.9628822720026312,  
2.0857180061833653, 1.0836210546074156, 2.050228476214192,  
1.3834202080435356, 2.146044835955279, 1.1856519908423175,  
1.3598623245075914, 2.038272187883535]
```

[Hide output](#)

**Note that we are not grading parts (a), (c), (e), and (h).**

[Submit](#)

You have used 1 of 100 attempts

✓ Correct (100/100 points)

© All Rights Reserved



© 2016 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

