CS 274A Homework 3

Probabilistic Learning: Theory and Algorithms, CS 274A, Winter 2022

Due: 11:59pm Thursday Feb 9th, submit via Gradescope

Instructions and Guidelines for Homeworks

- Please answer all of the questions and submit your solutions to Gradescope (either hand-written or typed are fine as long as the writing is legible).
- All problems are worth equal points (10 points) unless otherwise stated. All homeworks will get equal weight in computation of the final grade for the class (with lowest-scoring homework being dropped).
- The homeworks are intended to help you better understand the concepts we discuss in class. It is important that you solve the problems yourself to help you learn and reinforce the material from class. If you don't do the homeworks you will likely have difficulty in the exams later in the quarter.
- In problems that ask you to derive or prove a result you should submit a complete mathematical proof (i.e., each line must follow logically from the preceding one, without "hand-waving"). Be as clear as possible in explaining your notation and in stating your reasoning as you go from line to line.
- If you can't solve a problem, you can discuss the high-level concepts *verbally* with another student (e.g., what concepts from the lectures or notes or text are relevant to a problem). However, you should not discuss any of the details of a solution with another student. In particular note that you are not allowed to view (or show to any other student) *any written material* directly related to the homeworks, including other students' solutions or drafts of solutions, solutions from previous versions of this class, etc. The work you hand in should be your own original work.
- If you need to you can look up standard results/definition/identities from textbooks, class notes, textbooks, other reference material (e.g., from the Web). If you base any part of your solution on material that we did not discuss in class, or is not in the class notes, or is not a standard known result, then you may want to rovide a reference in terms of where the result is from, e.g., "based on material in Section 2.2 in ....." or a URL.

Recommended Reading for Homework 3: Note Set 4 and *Bayesian Learning/Recommended Reading* (links to Murphy notes and text) on the course Webpage under *Class Notes and links to additional reading.*

Problem 1: Beta-Binomial Model

Consider the binomial likelihood model with a beta prior as we discussed in class. Let the number of successes in the data be r out of n trials. Say the true unknown value of $\theta = 0.8$. Consider the following 2 data sets:

1. $r = 3, n = 3$;
2. $r = 36, n = 50$;

Consider also two possible sets of priors:

1. $\alpha = \beta = 1$
2. $\alpha = \beta = 10$.

1. Generate $2 \times 2 = 4$ plots on a single page, where each row corresponds to each of the 2 data sets and each column corresponds to the two priors. Plot the prior and the posterior densities for each case over the range $\theta \in [0, 1]$. Mark on the x-axis on your plot where the true value of $\theta = 0.8$ is located as well as the location of the maximum likelihood estimate. Clearly indicate on your plot what the different curves and marks correspond to. Also comment briefly on what you observe from these plots (a few sentences are fine).
2. A *posterior credible interval* in Bayesian analysis can be defined as a range for θ that contains most of the posterior probability mass. For example we could define a 95% posterior credible interval as the range θ corresponding to the region between $\theta \in [\theta_l, \theta_u]$ where θ_l and θ_u are defined as values to the left and right of which (respectively) are 2.5% of the "tail" probability mass. Using this definition of a credible interval, for each of the 4 combinations of priors and datasets determine what the values of θ_l and θ_u are in terms of defining a 95% posterior credible interval (no need to plot this, just state what this interval is for each of the 4 cases). Feel free to use whatever method you wish to determine θ_l and θ_u. Your answer should be accurate to within two decimal places.

You should generate your plots and credible intervals on a computer (rather than hand-drawn, etc), using whatever language you wish, but no need to submit any code.

Problem 2: Properties of the Beta Density

In class we discussed the use of a Beta density function as a prior density for a parameter that lies between 0 and 1. The Beta density is defined as:

$$P(\theta) \ = \ Be(\theta; \alpha, \beta) \ = \ \frac{1}{B(\alpha, \beta)} \theta^{\alpha - 1} (1 - \theta)^{\beta - 1} \tag{1}$$

where $0 \leq \theta \leq 1$. The two parameters of this density function are $\alpha > 0$ and $\beta > 0$ and $B(\alpha, \beta)$ is a normalization constant to ensure that the density integrates to 1, where $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$, $z > 0$, is the gamma function.

In solving the problems below keep in mind that since $B(\alpha, \beta)$ is the normalization constant for the density, then by definition $B(\alpha, \beta) = \int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta$. Another useful fact is that the gamma function has the property that $\Gamma(z+1) = z\Gamma(z), z > 0$.

1. Derive an expression for the mean of a Beta density, as a function of the parameters α and β.
2. Derive an expression for the mode of a Beta density, as a function of the parameters α and β. You can assume for this part of the problem that $\alpha > 1$ and $\beta > 1$.

Problem 3: Bayesian Estimation for the Multinomial Model

Consider a data set $D = \{x_1, \ldots, x_N\}$, with $x_i \in \{1, \ldots, K\}$ where the x_i's are conditionally independent draws from a discrete-valued distribution with parameters $\theta_k = P(x_i = k), 1 \leq k \leq K$, and $\sum_{k=1}^{K} \theta_k = 1$ (i.e., we have a multinomial likelihood for the x_i's). Assume that we have a Dirichlet prior for the parameters θ, where the prior has parameters $\alpha_1, \ldots, \alpha_K$ and $\alpha_k > 0$ and $1 \leq k \leq K$.

1. Prove that the posterior distribution on $\theta_1, \ldots, \theta_K$ is also Dirichlet and find it's parameters.
2. Derive an expression for the maximum a posteriori (MAP) estimate for $\theta_k, 1 \leq k \leq K$. Your solution should be derived from first principles (i.e. using basic calculus to find the mode of the posterior density for θ_k, working from your solution for $P(\theta|D)$ from part 1).

Problem 4: Bayesian Estimation for the Exponential Model

Consider a data set $D = \{x_1, \ldots, x_N\}$, where x_i's are real-valued and $x_i \geq 0$, and where the x_i's are conditionally independent draws from the exponential density $p(x|\theta) = \theta e^{-\theta x}, \theta > 0$.

Define a Gamma prior for θ in the form $p(\theta|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1}e^{-\beta\theta}$, where $\alpha > 0$ and $\beta > 0$ are the parameters of the Gamma prior and $\Gamma(.)$ is the Gamma function.

1. Derive an expression for the posterior density on θ and state what type of density this.
2. Derive an expression for the posterior mean and show that it can be written as a convex combination of the prior mean and the maximum likelihood estimate of θ.
3. Derive an expression for the posterior mode.

Problem 5: Bayesian Estimation for the Mean of a Gaussian Model

Consider a data set $D = \{x_1, \ldots, x_N\}$, with real-valued scalar x_i values. Assume that the x_i's are conditionally independent draws from a Gaussian density with unknown mean μ and known variance σ^2. Assume we have a Gaussian prior on μ that has mean μ_0 and variance s^2. Given the facts above, derive the following expressions from first principles, clearly explaining and justifying all of your steps:

$$\mu_N \ = \ \gamma \hat{\mu}_{ML} \ + \ (1-\gamma)\mu_0$$

and

$$\frac{1}{\sigma_N^2} = \frac{N}{\sigma^2} + \frac{1}{s^2}$$

where μ_N and σ_N^2 are the mean and variance of the posterior density for μ and

$$\gamma = \frac{Ns^2}{Ns^2 + \sigma^2}.$$

Problem 6: Bayesian Active Learning with MultiArm Bandits (30 points)

An active area of machine learning research is multi-arm bandits. In this problem we will see how Bayesian modeling can be useful in this context. We will only explore one small aspect of multi-arm bandit problems: there is a large literature on this topic in general, e.g., see recent machine learning conferences.

Consider a setup where we have K "bandits." You can think of each bandit as being a slot machine with an arm that we can pull. Whenever we pull a particular arm k (for bandit k) we get a stochastic reward $y_k \in \{0, 1\}$ that is drawn from some unknown Bernoulli distribution θ_k, where $\theta_k = P(y_k = 1)$, $k = 1, \ldots, K$.

The bandit problem we will investigate is how to sequentially explore the arms (i.e., select arms) to try to identify which of the arms (or bandits) has the highest expected reward, i.e., to identify $\arg\max_k\{\theta_k\}$. This problem shows up in a variety of real-world problems in areas like reinforcement learning, active learning, medical clinical trials, online advertising (learning which from a set of advertisements is the one that people are most likely to respond to), and so on.

In general we will consider N sequential trials indexed by $i = 1, \ldots, N$. At the start of each trial we select one arm k: we then "pull" that arm and we get a random y_k outcome (0 or 1) drawn from the true θ_k for the arm we pulled. In terms of notation, after we conduct trial i, $1 \le i \le N$ we will have 2 numbers for each arm. The first number is $n_k^{(i)}$, the number of times that arm k has been "pulled" (selected in a trial) up to and including trial i. The second number is $r_k^{(i)}$, the number of times that outcome $y_k = 1$ has been observed up to and including trial i. In general $0 \le r_k^{(i)} \le n_k^{(i)} \le i$.

We will investigate three different algorithms for selecting an arm k at each trial and we will explore how they differ:

1. Random: At each trial i this algorithm selects an arm uniformly at random among the K possibilities.

2. Greedy: At each trial i this algorithm selects the arm k that has the highest mean posterior estimate (MPE) across the K arms, i.e., select $\arg\max_k\{\theta_k^{MPE(i-1)}\}$ where each $\theta_k^{MPE(i-1)}$ is the MPE of θ_k for arm k based on the previous $i-1$ trials, i.e.,

$$\theta_k^{MPE(i-1)} = \frac{r_k^{(i-1)} + \alpha_k}{n_k^{(i-1)} + \alpha_k + \beta_k}$$

   where α_k and β_k are parameters of a Beta prior for each θ_k.

3. Thompson Sampling: This algorithm keeps track of a Beta posterior density for each arm k, which (before it selects an arm for trial i) will be based on the following numbers: $r_k^{(i-1)}, n_k^{(i-1)}, \alpha_k, \beta_k$. At the start of each trial i it then **samples θ_k values**, one from each of the K arms using the current posterior Beta densities for each arm. It then selects the arm k whose **sampled value** has the highest value among the K sampled values.

For simplicity in all the experiments in this problem we will use uniform (uninformative) prior values of $\alpha_k = \beta_k = 1, k = 1, \ldots, K$. If there are any ties in selection of an arm (e.g., this can often occur in early stages of the greedy algorithm), break ties randomly (this is important).

In general we would like to have an algorithm that can both (i) explore the arms (which Random will do, but Greedy won't do enough of), but (ii) also exploit the arms (by focusing on the more promising ones - which Random won't do, and Greedy will sometimes do too much of). Thompson Sampling provides a balance between exploring and exploiting. As you might imagine, the Greedy method can sometimes be quite suboptimal and since it can get "trapped" and not explore the set of arms enough. And it makes sense that there are more efficient strategies than Random.

The intuition for the sampling aspect of Thompson Sampling is that even if, at trial i, an arm only has a small probability of being the highest reward arm, there is still a chance of it being selected since it might possibly turn out to be the optimal arm in the long run once we see more data (hence, Thompson Sampling encourages some exploration, in a Bayesian fashion). The general idea of Thompson Sampling has some nice theoretical properties (see the discussion in Section 8 of the Russo et al tutorial).

For a more complete description of Thompson Sampling for Bernoulli bandits (which is the problem we described above) please read Sections 1 through 3 of the following tutorial by Russo et al on Thompson Sampling: `https://arxiv.org/pdf/1707.02038.pdf`. Theoretical properties are discussed in Section 8.

Your goal is to implement in code (in any language you wish) the three algorithms, Greedy, Random, and Thompson Sampling and then evaluate these methods using the instructions below.

Experiments across Multiple Runs

You will implement the 3 algorithms and explore how they work on simulated problems. A "run" consists of N trials: each run can produce different results since the y_k values are stochastic for each trial i, and in addition the Random and Thompson Sampling algorithms have randomness built into how they select arms to pull.

Thus, any single run can be quite noisy due to the stochasticity of the problem. So, to average over this stochastic noise (in terms of evaluating systematic differences between algorithms), you will generate M runs for each algorithm (independently, with different random outcomes for the selected arms each time, and different random selections by Random and Thompson Sampling at each trial), where each run is of length N. Thus, for each of the 3 algorithms you will end up with M runs each with N trials, and sets of numbers $n_k^{(i)}$ and $r_k^{(i)}$, $k = 1, \ldots, K$ and $i = 1, \ldots, N$ for each trial.

To evaluate how good an algorithm is on average after i trials, you can compute the average value (across M runs) of its performance after each trial $i = 1, \ldots, N$. The performance metric we will use is the success rate, i.e., the fraction of times (across runs) that an algorithm's estimate of the best arm $\hat{k}^{(i)}$, after i trials, is the same as the true best arm $k^* = \arg\max_k \theta_k$, where the θ_k's are the true parameters for each arm (and are not known to the algorithms). At the end of each trial i, each of the 3 algorithms can use its own estimate of $\arg\max_k\{\theta_k^{MPE}\}$ to select the best arm, where θ_k^{MPE} may in general be different for each of the algorithms since they have each collected different data, i.e., will have different values for $n_k^{(i)}$ and $r_k^{(i)}$, $k = 1, \ldots, K$. Note: for evaluation purposes (selecting the best arm) we don't do any additional sampling with Thompson Sampling at trial i: we just want to know how accurate it would have been in identifying the best arm, on average across the M runs, if we had stopped after i trials.

You can then plot (on the y-axis) the fraction of correct identifications (on average across M runs), for each value of $i = 1, \ldots, N$ (on the x-axis), for each of the 3 algorithms. The y-axis will range from 0 to 1. The curves will generally start at low values (around $1/K$) for small values of i and then increase (at least for Thompson Sampling and Random) towards 1. The rate at which they approach 1 will depend on the values of θ_k's and will differ between strategies. Greedy will tend to be the noisiest and least predictable across runs.

What to Submit:

Submit the following plots for this problem. Use $M = 500$ for all your plots and generate 3 graphs, one for each setting of the θ's below, where each graph has 3 curves (one per algorithm) and the x-axis runs from 1 to N. (Feel free to make M much larger if you want to get smoother plots and if your code runs fast enough to do more runs). **Please submit your code** in your programming language of choice at the end of your solution file.

- Let $K = 10$ with $\theta_1 = 0.9, \theta_2 = 0.8, \theta_3 = \theta_4, \ldots = \theta_{10} = 0.5$. For $N = 1000$ trials plot the success rate for each of the three strategies.
- Repeat part 1 (i.e., generate another plot) but now with $\theta_1 = 0.9, \theta_2 = 0.88, \theta_3 = \theta_4, \ldots = \theta_{10} = 0.5$. Set $N = 3000$ trials.
- Repeat part 1 (i.e., generate another plot) but now with $\theta_1 = 0.9, \theta_2 = \theta_3 = \theta_4, \ldots = \theta_{10} = 0.85$. Set $N = 3000$ trials.

- Comment on (i) the differences in results between the 3 algorithms for each plot, and (ii) the differences in results in general between the 3 plots as the true θ's change.
- Optional: (no extra credit but fun to think about): for the random strategy, can you think of how you might analytically compute the success rate as a function of i (rather than simulating it) if you knew the θ's. You could just consider the special case of $K = 2$ bandits. What insights can you gain from your analysis? i.e., in terms of how hard or easy a particular identification problem is.

Note that you should be able to get your code to run fairly quickly, e.g., in less than a minute for each of the plots being requested, if your code is reasonably efficient. You may want to make sure your code is working first (e.g., try an "easy" problem first for debugging with just $K = 2$ arms and a large difference in the theta values).

Finally, as a sidenote for those of you who may be interested, here is a pointer to a paper in 2021 from my research group where we used ideas from Bayesian estimation, multi-arm bandits, and Thompson sampling to address the problem of assessing the accuracy of black-box classification models `https://arxiv.org/pdf/2002.06532.pdf` (Ji et al, AAAI Conference, 2021).