CS 274A Homework 3

Probabilistic Learning: Theory and Algorithms, CS 274A, Winter 2020

Due Date: Noon Wednesday Feb 4th, submit via Gradescope

Instructions and Guidelines for Homeworks

- Please answer all of the questions and submit a scanned copy of your written solutions to Gradescope (either hand-written or typed are fine as long as the writing is legible).
- All problems are worth 10 points unless otherwise stated. All homeworks will get equal weight in computation of the final grade for the class.
- The homeworks are intended to help you work through the concepts we discuss in class in more detail. It is important that you try to solve the problems yourself. The homework problems are important to help you better 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.
- If you can't solve a problem, you can discuss it *verbally* with another student. However, please note that before you submit your homework solutions 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, and so forth. The work you hand in should be your own original work.
- You are allowed to use reference materials in your solutions, such as class notes, textbooks, other reference material (e.g., from the Web), or solutions to other problems in the homework. It is strongly recommended that you first try to solve the problem yourself, without resorting to looking up solutions elsewhere. If you base your solution on material that we did not discuss in class, or is not in the class notes, then you need to clearly provide a reference, e.g., "based on material in Section 2.2 in ....."
- In problems that ask for a proof 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 wish to use LaTeX to write up your solutions you may find it useful to use the .tex file for this homework that is posted on the Web page. And please feel free to submit the .tex (as well as the .pdf) file for your solutions —it may be helpful to us when we send out solutions later on.

Suggested reading for Homework 3:

- Notes on class Web page on Bayesian estimation, particularly the instructor notes and notes by Kevin Murphy.

Problem 1: 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 that the gamma function has the property that $\Gamma(z+1) = z\Gamma(z)$.

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 2: 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 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.
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 3: Bayesian Estimation of a Gaussian Model

Consider a data set $D = \{x_1, \ldots, x_N\}$, with real-valued scalar x_i. 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 with a

Gaussian prior on μ that has mean μ_0 and variance s^2. As discussed in class the posterior density for μ is also Gaussian with parameters μ_N and σ_N^2. Given the facts above, derive the following expressions:

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

and

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

where

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

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 $x_i \geq 0, 1 \leq i \leq N$, and where the x_i's are independent draws from the exponential density $p(x|\theta) = \theta e^{-\theta x}$.

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 special Gamma function. Note that when $\alpha = 1$ we get an exponential density (on θ, with parameter β) as a special case.

1. Derive an expression for the posterior density on θ and state what type of density this.
2. What are the posterior mode and posterior mean for this model? For this problem you can state the results, you don't need to derive the posterior mean and mode from first principles unless you want to.

Problem 5: Simulation of Beta Densities

Write code in your language of choice to do the following. Submit the 3 figures (with 9 plots in each figure) as requested below. No need to submit the code. Also comment briefly on what can be learned from these plots (a few sentences are fine).

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. Consider the following 3 data sets:

1. $r = 2, n = 5$;
2. $r = 8, n = 20$;
3. $r = 40, n = 100$;

For each of the above data sets you are to generate a single figure (to fit on one page), where each figure contains 9 different plots arranged in 3 rows and 3 columns. The plots in rows 1, 2, and 3 should correspond to "prior strengths" of $\alpha + \beta = 1, 10, 100$ respectively. The plots in columns 1, 2, and 3 should correspond to prior means of 0.2, 0.4, and 0.8 respectively. For each individual plot, plot the prior $p(\theta)$ and the posterior $p(\theta|D)$. The prior should be plotted using a dotted line, the posterior using a solid line—use red and green colors (respectively) for each line. In summary, you will be generating 3 separate figures, where each contains $3 \times 3 = 9$ plots.

Problem 6: Bayesian Active Learning with MultiArm Bandits

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. When we pull arm k we get a stochastic reward $y_k \in \{0, 1\}$. Each bandit has its own (unknown) Bernoulli distribution θ_k, where $\theta_k = P(y_k = 1)$. 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 many practical settings such as online advertising (which advertisement is the one that people are most likely to click on) and medical clinical trials (which drug works best on average).

Your goal is to implement in code (in any language you wish) an algorithm called Thompson Sampling for this problem and compare it to two baselines: (1) Random, where you just select (and play) arms in random order, and (2) Greedy, where you always play whatever your current estimate of the best arm is. As you might imagine, the Greedy method can be quite suboptimal and get "trapped" since it doesn't explore the space enough. In general we want 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 do too much of). Thompson Sampling provides a balance between exploring and exploiting.

For a description of Thompson Sampling for Bernoulli bandits (which is the problem we described above) please read Sections 1 through 3 of this tutorial by Russo et al on Thompson Sampling: `https://arxiv.org/pdf/1707.02038.pdf`.

One run of each of the 3 algorithms will consist of N trials (each trial corresponds to pulling one of the K arms). At trial i, each algorithm selects an arm to pull, call it $k^{(i)}$. It pulls that arm and gets a reward $y^{(i)} \in \{0, 1\}$. It then updates its record for arm k where n_k is incremented by 1 (number of times arm k has been pulled) and r_k is incremented by 1 if $y^{(i)} = 1$.

The Random and the Greedy strategy keep track of maximum likelihood estimates $\theta_k^{ML} = \frac{r_k}{n_k}$ for each arm. Thompson Sampling (Algorithm 2 in the Russo et al tutorial above) updates its posterior Beta density for each arm, updating n_k (and possibly r_k) in the parameters for its Beta density for θ_k (for arm k). (Note that the Russo et al paper has a different variation for a Greedy (that they call Algorithm 1) that picks the k with the largest MPE at each trial (rather than using the ML estimate): we are not using this version but its performance should be very similar to (or even the same as, for uniform priors) our version which uses the ML estimates).

At the next trial $i + 1$, each strategy selects an arm to pull in a different way. Random selects an arm randomly from the K options (each arm equally likely to be selected). Greedy selects the arm that currently has the highest value of $\theta_k^{ML} = \frac{r_k}{n_k}$. Thompson Sampling is different: it samples a θ_k value from each of the K current Beta densities on the K arms, and it then selects **the sample** with the highest value. The heuristic here is that even if an arm only has a small probability of being the highest reward arm, there is still a chance of it being selected. The general idea of Thompson Sampling has some nice theoretical properties (see the discussion in Section 8 of the Russo et al tutorial).

You will implement the 3 strategies and see explore how they work on simple problems. One "run" consists of running a strategy to select arms (and update its statistics) for $i = 1, \ldots, N$ consecutive trials. These runs can be quite noisy due to the stochasticity of the problem: so you will need to generate M such runs (independently, with different random outcomes each time), each of length N, to average over the noisiness of each run. Thus, for each strategy you will end up with M runs of length N.

To evaluate a strategy you can compute the average value (across the M runs) of some performance metric for each value $i = 1, \ldots, N$. The performance metric we will use is the success rate, i.e., the fraction of times that a strategy had identified the true best arm (largest θ_k) after i trials. For Greedy and Random, their estimate of the best arm is $\arg\max_k\{\theta_k^{ML}\} = \arg\max_k\{\frac{r_k}{n_k}\}$, at the ith trial, based on the arms they have explored up and including trial i. For Thompson sampling, we can use the MPE estimate, i.e., $\arg\max_k\{\theta_k^{MPE}\}$. Note that for evaluation we don't do any sampling: we want to know what would Thompson sampling have given us, on average across the M runs, if we had stopped after i trials.

You can then plot the fraction of correct identifications (on average across M runs), for each value of $i = 1, \ldots, N$, for each of the 3 strategies. The curves will generally start at low values for small values of i and then gradually increase towards 1. The rate at which they approach 1 will depend on the values of θ_ks and will differ between strategies.

Submit the following for this problem:

- Let $K = 10$ with $\theta_1 = 0.9, \theta_2 = 0.8, \theta_3 = \theta_4, \ldots = \theta_{10} = 0.5$. For $N = 500$ trials and $M = 500$ runs, plot the success rate for each of the three strategies.
- Repeat part 1 (i.e., generate another plot) but now with $\theta_1 = 0.92, \theta_2 = 0.9, \theta_3 = \theta_4, \ldots = \theta_{10} = 0.5$. Set $N = 4000$ trials and (if you can) make $M = 500$.
- Comment on (i) the differences in results between the three strategies across both plots, and (ii) the differences in results in general between the 2 plots from part 1 and part 2.
- (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, if you knew the θ's (rather than simulating it). 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.

For both parts 1 and 2 feel free to make M larger if you wish (larger M will smooth out more of the stochastic noise across runs).

In the experiments above the prior (for Thompson Sampling) for each arm k should be a Beta density with $\alpha = \beta = 1$ (i.e., a relatively weak uninformative prior). If there are any ties in terms of selecting the best arm on a particular trial (e.g., for the Greedy strategy), ties should be broken randomly.

Note that you should be able to get your code to run quickly (e.g., a few seconds to generate the graphs across all trials and all runs in parts 1 and 2 above). If you have any tips to share on speeding these simulations up for a specific language (like Python, R, etc) feel free to share them on Piazza.

You do not need to submit any code, just the plots. If you think your code is potentially useful to others and you are willing to share it (and whether its written in R, Python, C, Matlab, or something else), feel free to email it to the instructor as Zip file and we can make it available to students who might be interested in running it after all homeworks are submitted.