CS 274A Homework 1

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

Due: 12 noon Thursday January 16th, 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 1:

- If you need to brush up on your knowledge of basic probability, reading Note Sets 1 and 2 from the class Web page may be helpful
- Chapter 6 in *Mathematics for Machine Learning* contains additional depth on basic concepts in probability, some of which may be helpful in the problems below. Section 8.5 on graphical models in Chapter 8 of MML may also provide useful background for the final problem.

Problem 1: Expectations/Variance with Two Random Variables

The expected value of a real-valued random variable X, taking values x, is defined as $\mu_x = E[X] = \int p(x)\ x\ dx$ where $p(x)$ is the probability density function for X. The variance is defined as $var(X) = E[(X - \mu_x)^2] = \int p(x)(x - \mu_x)^2 dx$ (often also denoted as σ_x^2). In the questions below a and b are scalar constants (i.e., not random variables).

Let X and Y be two real-valued random variables. In the equations below the expectation on the left is with respect to the joint density $p(x, y)$ and the expectations on the right are with respect to $p(x)$ and $p(y)$ respectively.

1. Prove that $E[aX + bY] = aE[X] + bE[Y]$.
2. Prove that if X and Y are independent that $var(aX + bY) = a^2 var(X) + b^2 var(Y)$.

Problem 2: Uniform Density

Let X be a continuous random variable with uniform density $U(a, b)$, with $a < b$, i.e.,

$$p(x) \ = \ p(X = x) \ = \frac{1}{b - a}$$

if $a \leq x \leq b$ and $P(x) = 0$ otherwise.

1. Derive an expression for $E[X]$.
2. Derive an expression for $var(X)$.

Problem 3: Geometric Model

Suppose that we repeatedly toss a coin (with no memory in the coin, so that tosses are independent) until we get the outcome of "heads." Let θ be the probability of heads on any toss. Let X be the number of such tosses until a heads occurs. This type of model can be used to describe "independent trials" and is frequently used in science and engineering to model various simple repetitive phenomena (e.g., how many times one uses a device until it breaks, the number of consecutive days of rainfall at a given location, and so on).

In this situation X is a discrete random variable with a geometric distribution taking values $x \in \{1, 2, 3, \ldots, \}$, with a probability distribution defined as $P(x) = (1-\theta)^{x-1}\theta$. Here θ is the parameter of the geometric model (e.g., the probability of heads in coin-tossing) and $0 < \theta < 1$.

1. Prove that $\sum_{x=1}^{\infty} P(x) = 1$.
2. Derive an expression for the expected value of X, $\mu_X = E[X]$.
3. Derive an expression for the variance of X, where the variance is $\sigma_x^2 = E[(X - \mu_x)^2]$.

Problem 4: Central Limit Theorem

Let $X_1, \ldots, X_n$ be a set of independent and identically distributed real-valued random variables each with the same density $P(X)$ where $P(X)$ has mean μ and variance σ^2.

1. State the central limit theorem as it applies to $X_1, \ldots, X_n$ (if you don't know the what central limit theorem is you will need to look it up)
2. Let $Y = \frac{1}{n}\sum_{i=1}^{n} X_i$ where each X_i has a uniform distribution between 0 and 1. Simulate 1000 values of Y for each of the following values of n, $n = 10^2, 10^3, 10^4, 10^5$. (So you should end up with 4 sets of Y values, each with 1000 values). For example you can use the `rand.m` function in Matlab to do this, or similar functions in R or Python. Generate histogram plots of the 4 results (e.g., using the `hist.m` function in MATLAB) for each value of n (this will produce 4 histograms). Use $\sqrt{1000} \approx 30$ bins in your histograms.
3. Based on visual inspection of the histograms, comment on how your simulated data matches the central limit theorem.
4. Quantitatively evaluate how well your empirically simulated distributions match what the theory predicts (e.g., compare the mean and variance of the simulated data with that from theory).

Problem 5: Finite Mixture Models

Finite mixture models show up in a wide variety of contexts in machine learning and statistics (we will discuss them in more detail in lectures later in the quarter). In this problem consider a real-valued random variable X taking values x (in general we can define mixtures on vectors, but here we will just consider the 1-dimensional scalar case). The basic idea is to define a density (or distribution) $p(x)$ that is a weighted mixture of K component densities $p_k(x)$ where the weights are non-negative and sum to 1, i.e.,

$$p(x) \; = \; \sum_{k=1}^{K} \alpha_k \; p_k(x; \theta_k)$$

where

- the weights obey the following conditions: $\sum_{k=1}^{K} \alpha_k = 1, \;\; 0 \leq \alpha_k \leq 1$
- each $p_k(x; \theta_k)$ is itself a probability density function with its own parameters θ_k. For example, if a component is Gaussian then $\theta_k = \{\mu_k, \sigma_k^2\}$.

The full set of parameters for a mixture model consists of both (a) the K weights, and (b) the K sets of component parameters θ_k for each of the K mixture components. (Note that the "finite" in finite mixture models comes from the fact that K is finite. There are also infinite mixture models where K is unbounded, but we will not consider those here).

1. Given the properties above prove that a finite mixture $p(x)$ is itself a density function, i.e., it obeys all the necessary properties needed to be a density function.
2. Derive general expressions for the (a) mean μ of $p(x)$, and (b) the variance σ^2 of $p(x)$, as a function of the component weights, means and variances $\alpha_k, \mu_k, \sigma_k^2, 1 \leq k \leq K$. For each of μ and σ^2 provide an intuitive interpretation in words of your equations.

Problem 6: Multivariate Gaussians

Ad defined in Note Set 2, the joint multivariate (multidimensional) Gaussian density, for a d-dimensional vector of real-valued random variables $X_1, \ldots, X_d$, is defined as:

$$p(\underline{x}) = p(x_1, x_2, \ldots, x_d) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu})^T \Sigma^{-1} (\underline{x}-\underline{\mu})}$$

where $\underline{x}$ is a d-dimensional vector (a set of possible values for $X_1, \ldots, X_d$), $\underline{\mu} = (\mu_1, \ldots, \mu_d)$ is a $d \times 1$ dimensional vector of mean values and Σ is a $d \times d$ positive semi-definite symmetric covariance matrix with entries $\sigma_{ij} = \sigma_{ji} = cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)]$ with $1 \leq i, j \leq d$. The diagonal terms $\sigma_{ii} = cov(X_i, X_i)$ are the variances $\sigma_i^2 > 0$ for each of the individual X_i variables.

In the two-dimensional case, $d = 2$, with $\underline{x} = (x_1, x_2)$ we can write the covariance matrix in a form such that $\sigma_{12} = \sigma_{21} = r\sigma_1\sigma_2$, where σ_1 and σ_2 are the standard deviations of X_1 and X_2 respectively and where r is the well-known linear correlation coefficient with $-1 < r < 1$. (We will keep $|r| < 1$ to avoid degenerate cases).

1. Prove that in two dimensions that any isocontour of a Gaussian PDF is an ellipse in the general case. An isocontour is defined by $p(\underline{x}) = c$, where $c > 0$ is some scalar constant and c is less than the maximum value of $p(\underline{x})$.
2. For $\underline{\mu} = [0, 0], \sigma_1 = 1, \sigma_2 = 1$, for each of the three cases $r = 0.9, r = 0, r = -0.9$, write code (in whatever language you wish) to plot (in 2d) the isocontours that correspond to c taking values $\alpha \; c_{\max}$ where α takes values such as $[0.9, 0.7, 0.5, 0.3, 0.1]$ and $c_{\max}$ is the value of $p(\underline{x})$ at $\underline{\mu}$ (i.e., at the mode). Submit your 3 plots (no need to submit your code) and clearly label the units on each axis.

Problem 7: Simulation of Multivariate Gaussian Data

In this problem you will look at how one can simulate independent samples from a multivariate Gaussian (or Normal) density.

Assume that we have access to a function that can generate pseudo-random uniformly-distributed samples from the $U(0,1)$ density. Say we wish to generate a random sample $\underline{x}$ from a Normal (Gaussian) density with d-dimensional mean vector $\underline{\mu}$ and $d \times d$ covariance matrix Σ. One way to do this is the following:

- generate a vector of d independent univariate normal random variables, each with mean 0 and variance σ^2 (call this vector $\underline{z}$). This vector can be generated using (a) a function that generates pseudo-random uniformly-distributed samples from the $U(0,1)$ density, and (b) the Box-Muller method that transforms each $U(0,1)$ sample into a Normal/Gaussian $N(0,1)$ sample;
- generate a linear transform $\underline{x} = A\underline{z} + \underline{\mu}$ of $\underline{z}$, where $AA^T = \Sigma$, where A^T is the transpose of A, and where we can compute A from Σ using the Cholesky decomposition method. One can show that the resulting $\underline{x}$ will be a pseudorandom sample from the multivariate Normal $N(\underline{\mu}, \Sigma)$.

You will need to look-up the Box-Mueller and Cholesky decomposition methods and read about them (e.g., via Wikipedia) if you are not familiar with them.

Answer the following questions:

1. Write pseudocode that takes as input $\underline{\mu}, \Sigma$ and N, and returns N pseudorandom numbers sampled from $N(\underline{\mu}, \Sigma)$, using the Box-Muller transformation and the Cholesky decomposition as described above. Your pseudocode can call functions called `box-muller()` that transforms two uniform random numbers into two normally distributed ones, and `cholesky()` that takes Σ as input and returns A.
2. Determine the time complexity ("big O" notation) of your pseudocode as a function of N and d. You can treat multiplications, divisions, trigonometric function calls, generation of a uniform pseudo-random number, etc, as all having complexity $O(1)$.
3. Prove that the linear transformation part of the method gives us valid multivariate normal samples, i.e., that if the elements of $\underline{z}$ are independent $N(0,1)$ random variables, then $\underline{x} = A\underline{z} + \underline{\mu}$ has a $N(\underline{\mu}, \Sigma)$ distribution, where $AA^T = \Sigma$.

Problem 8: Logistic Function

Let X be a d-dimensional real-valued random variable taking values $\underline{x}$ and let C be a binary random variable. Say we want to define the conditional probability $P(C = 1|\underline{x})$ as a function of $\underline{x}$. One well-known approach is to assume that this is the logistic function (the basis of the logistic regression classifier), defined as

$$P(C = 1|\underline{x}) \;=\; \frac{1}{1 + exp(-\alpha_0 - \alpha^T \underline{x})}$$

where α_0 is a real-valued scalar and α^T is the transpose of a $d \times 1$ vector of real-valued coefficients $\alpha_1, \ldots, \alpha_d$. In machine learning C is typically referred to as the "class," in this context: its the variable we want to predict given $\underline{x}$.

- Prove that the definition of the logistic function above is equivalent to assuming that the log-odds $\log \frac{P(C=1|\underline{x})}{P(C=2|\underline{x})}$ is linear in $\underline{x}$.
- Say we know that $P(\underline{x}|C=1) = N(\underline{\mu}_1, \Sigma)$ and $P(\underline{x}|C=2) = N(\underline{\mu}_2, \Sigma)$ where $\underline{\mu}_1$ and $\underline{\mu}_2$ are the d-dimensional means for each class and Σ is a common covariance matrix. Prove that, under these assumptions, $P(C=1|\underline{x})$ is in the form of a logistic function. (Hint: you may find the algebra to be easier if you utilize the result from part 1).

Problem 9: Graphical Models

Consider a directed graphical model with random variables A, B, C, D, E, F where F has parent E, E has parent C, D has parent B, C has parents A and B, and A and B each have no parents. Assume that each variable can take K values, $K \geq 2$.

1. Draw a diagram showing the structure of this graphical model and write down an expression for the joint distribution $P(a, b, c, d, e, f)$ as represented by this graphical model.
2. Specify precisely how many parameters are in this graphical model. A "parameter" here is defined as any conditional probability or marginal (unconditional) probability that is needed in any probability table required to specify the model.
3. How many parameters would be required if we had a fully saturated model? (i.e., a model where no conditional independencies assumed).
4. Use both (a) the law of total probability, and (b) the structure of the graphical model, to show (in a series of equations) how one could use the structure of the model to efficiently compute the conditional distribution $p(f_k|a^*, d^*)$ for all values f_k, where a^* and d^* are some fixed observed values for A and D and the other variables are unobserved. Some hints on how to tackle this question:
    - The hints below focus on first computing the joint probability $p(f_k, a^*, d^*)$ for some specific value of f_k. You can then repeat this K times for each possible value f_k, and then normalize these joint probabilities to sum to 1 (a computation with complexity $O(K)$) to get the conditional probabilities $p(f_k|a^*, d^*)$.
    - You should start with the law of total probability with $p(f_k, a^*, d^*)$ on the left-hand side and a sum over the unknown variables on the right, and then replace the full joint distribution with the factorization implied by the graph.
    - From a computational efficiency perspective, its best to decompose the problem and to use the structure of the factorization in the model to "work your way down the graph," i.e., start by computing the table $p(c_k, a^*, d^*)$, and so on.

- The computations should be expressed as functions of the conditional probabilities in the tables in the model.

5. What is the computational complexity ("big O") of the computation of $p(f_k|a^*, d^*)$ by (a) using your equations above, and (b) if you computed $p(f_k|a^*, d^*)$ for a saturated model (i.e., a model with no conditional independence structure).

6. What is the computational complexity, using the structure of the graphical model, if now A is also unknown, i.e., we just want $p(f|d^*)$? (no need to write out the equations, you can just state the complexity).