Homework 3: Continuous Variables & Classification

UC Irvine CS177: Applications of Probability in Computer Science

Due on October 31, 2019 at 11:59pm

Question 1: (20 points)

The time that a TA spends helping an individual student in office hours is exponentially distributed with a mean of 8 minutes, and independent of the time spent with other students. For an exponential distribution with parameter θ,

$$f_X(x) = \theta e^{-\theta x} \text{ for } x \geq 0, \qquad E[X] = \frac{1}{\theta}, \qquad \text{Var}[X] = \frac{1}{\theta^2}.$$

Suppose there is a homework due tomorrow, and there are 4 people ahead of you in line.

a) *The total time that it takes all 4 students ahead of you to receive help from the TA is a random variable. What is the mean of this total time?*

b) *What is the standard deviation of the total time taken by the 4 students ahead of you?*

c) *What is the probability that all 4 people ahead of you will each take at most 10 minutes?*

d) *Suppose that the TA has finished helping the first 3 students, and has already spent 10 minutes with the fourth student. What is the mean and standard deviation of the amount of additional time you need to wait for help?*

Question 2: (20 points)

An artificial intelligence class has an assignment to write a program that generates the next move in a game of chess. Suppose that the runtimes of student programs follow a normal distribution with mean $\mu = 13$ seconds, and standard deviation $\sigma = 2.0$ seconds. **Hint:** The Python commands `scipy.stats.norm.cdf` and `scipy.stats.norm.ppf` may be useful.

a) *What is the probability that a random program has a runtime greater than 18 seconds?*

b) *What is the probability that a random program has a runtime between 10 and 16 seconds?*

c) *The TA's want to help the students complete their work faster. What would they have to lower the average runtime to so that only 1.0% of students have runtimes over 13 seconds? Assume the standard deviation remains fixed at $\sigma = 2.0$ seconds.*

Question 3: (20 points)

You've been asked to test the performance of a batch of newly fabricated processors. If the processors were correctly manufactured (class $Y = 0$), the time X to complete your test suite is exponentially distributed with mean 1. If the equipment at the factory malfunctions (class $Y = 1$), the time X is exponentially distributed with mean 50. You must decide whether or not this batch of processors was correctly manufactured.

For the scenarios in the three parts below, it is possible to show that the optimal Bayesian classifier predicts $Y = 0$ if $x \leq c$, and predicts $Y = 1$ if $x > c$, for some constant c. The value of c depends on the test time distributions, the prior probabilities of the two classes, and the assumed loss function. You need to determine the optimal c in each case.

a) Suppose that a new fabrication process has just been deployed, and the probability that the factory manufactures correctly functioning processors is only $P(Y = 0) = 0.5$. What threshold c of the observed test suite time $X = x$ maximizes the probability that your prediction is correct?

b) Suppose that after some improvements to the new fabrication process, the probability that the factory manufactures correctly functioning processors increases to $P(Y = 0) = 0.99$. What threshold c of the observed test suite time $X = x$ maximizes the probability that your prediction is correct?

c) Market research suggests that the loss (or cost) of a missed detection (predicting $Y = 0$ when the processor is actually defective) is 500 times greater than the loss of a false alarm (predicting $Y = 1$ when the processor was correctly manufactured). Assuming again that $P(Y = 0) = 0.99$, what threshold c of the observed test suite time $X = x$ minimizes the expected loss?

Question 4: (40 points)

For a given day i, we let $Y_i = 1$ if the ground-level ozone concentration near some city (Houston, in our data) is at a dangerously high level. This is called an "ozone day". We let $Y_i = 0$ if the ozone concentration is low enough to be considered safe.

We want to predict Y_i from more easily measured "features" describing atmospheric pollutant levels and meteorological conditions (temperature, humidity, wind speed, etc.). There are a total of $M = 72$ of these features collected each day, which we denote by $X_i = \{X_{ij} \mid j = 1, \ldots, M\}$. Each feature $X_{ij} \in \mathbb{R}$ is a real number, and we will thus use a Gaussian distribution to model these continuous random variables.

We will build a "naive Bayes" classifier, which predicts observation i to be an ozone day if $P(Y_i = 1 \mid X_i) > P(Y_i = 0 \mid X_i)$, and a non-ozone day otherwise. Using Bayes rule, this classifier is equivalent to one that chooses $Y_i = 1$ if and only if

$$\frac{p_Y(1) f_{X|Y}(x_i \mid 1)}{f_X(x_i)} > \frac{p_Y(0) f_{X|Y}(x_i \mid 0)}{f_X(x_i)},$$

$$\ln p_Y(1) + \ln f_{X|Y}(x_i \mid 1) > \ln p_Y(0) + \ln f_{X|Y}(x_i \mid 0). \tag{1}$$

In this equation, $p_Y(y_i)$ is the probability mass function that defines the prior probability of ozone and non-ozone days. The conditional probability density function $f_{X|Y}(x_i \mid y_i)$ describes the distribution of the $M = 72$ environmental features, which we assume depends on the type of day. We make two simplifying assumptions about these densities: the features X_{ij} are conditionally independent given Y_i, and their distributions are Gaussian. Thus:

$$f_{X|Y}(x_i \mid 1) = \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{1j}^2}} \exp\left\{-\frac{(x_{ij} - \mu_{1j})^2}{2\sigma_{1j}^2}\right\}, \tag{2}$$

$$f_{X|Y}(x_i \mid 0) = \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{0j}^2}} \exp\left\{-\frac{(x_{ij} - \mu_{0j})^2}{2\sigma_{0j}^2}\right\}. \tag{3}$$

Given $Y_i = 1$, X_{ij} is Gaussian with mean μ_{1j} and variance σ_{1j}^2. Given $Y_i = 0$, X_{ij} is Gaussian with mean μ_{0j} and variance σ_{0j}^2. There are a total of $2M$ mean parameters and $2M$ variance parameters, since every feature X_{ij} has a distinct distribution for each of the two classes.

a) Derive equations for $\ln f_{X|Y}(x_i \mid 1)$ *and* $\ln f_{X|Y}(x_i \mid 0)$*, the (natural) logarithms of the conditional probability density functions in Equations (2,3). For numerical robustness, simplify your answer so that it does* not *involve the exponential function.*

Because ozone days are relatively rare, a classifier that always predicts $Y_i = 0$ would be correct over 95% of the time, but would obviously not be practically useful for reducing ozone hazard. To evaluate our classifiers, we will thus separately compute the numbers of *false alarms* (predictions of ozone days when in reality $Y_i = 0$) and *missed detections* (predictions of non-ozone days when in reality $Y_i = 1$). We are willing to allow some false alarms as long as there are very few missed detections.

For all parts below, assume that the mean parameters μ_{1j}, μ_{0j} are set to match the mean of the empirical distribution of the training data. The demo code computes these means.

b) Start by assuming the classes are equally probable ($p_Y(1) = p_Y(0) = 1/2$), and have unit variance ($\sigma_{1j}^2 = \sigma_{0j}^2 = 1$). Write code to compute the log conditional densities from part (a). Then using Equation (1)*, classify each test example. Report your classification accuracy, and the numbers of false alarms and missed detections.*
Hint: Your classifer should have fewer than 10 missed detections.

c) Rather than assuming features have variance one, set the variance parameters $\sigma_{1j}^2, \sigma_{0j}^2$ equal to the variance of the empirical distribution of the training data. Classify each test example using Equation (1) *with these variance estimates. Report your classification accuracy, and the numbers of false alarms and missed detections.*

d) Rather than assuming the classes are equally probable, estimate $p_Y(1)$ as the fraction of training examples that are ozone days. Classify each test example using Equation (1) *with this informative class prior, and the variances from part (c). Report your classification accuracy, and the numbers of false alarms and missed detections.*