Homework 3: Conditional Probability

January 26, 2025

Zifeng Wang
CSE 312(12973)/Section B

---

1. An Independent Beagle (9 points)

Ans: Gradescope Question

2. It's Pretty Bayesic (6 points)

Ans: Gradescope Question

3. ℙNA (15 points)

You know that Xena's parents have type A blood, but Xena's sister Yvonne has type O.

(a) What are the genotypes of Yvonne and their two parents, from the information above?

(b) With that information in mind, what is the probability that Xena carries an O gene, supposing that Xena has type A blood?

(c) Xena marries Zachary, who has type O blood. You may consider Zachary's blood type to be part of the sample space.
Compute the probability that Xena and Zachary's first child will have type O blood, still supposing that Xena has type-A blood.

(d) Suppose that Xena and Zachary's first child had type A blood and Xena has type-A blood; conditioned on both of those pieces of information, what is the probability that Xena carries an O gene?

Ans:

(a) Yvonne's genotype: Since Yvonne has type O blood, her genotype must be OO (phenotype O requires two O alleles, as O is recessive).
Genotypes of the parents: Both parents have type A blood. For their child (Yvonne) to have genotype OO, each parent must carry at least one O allele. Therefore, the genotypes of the parents must be AO and AO.

(b) Step 1: Define the sample space
Xena's parents have genotypes AO and AO. Each parent can pass down either an A or an O allele to Xena, with equal probability. This creates the following possibilities for Xena's genotype({AA, AO, OA, OO}):
$P(G_x = AA) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ (A from both parents).
$P(G_x = AO) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}$ (A from one parent, O from the other).
$P(G_x = OO) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ (O from both parents).
Step 2: Using the definition of conditional probability:

$$P(G_X = AO|Ph_X = A) = \frac{P(G_X = AO \cap Ph_X = A)}{P(Ph_X = A)}$$

If Xena's genotype is AO, her phenotype must be A. Thus,$P(G_X = AO \cap Ph_X = A) = P(G_X = AO) = \frac{1}{2}$
The probability that Xena has type A blood is the sum of probabilities for all genotypes consistent with phenotype A: $P(Ph_X = A) = P(Ph_X = AA) + P(Ph_X = AO) = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$.
Thus: $P(G_X = AO|Ph_X = A) = \frac{\frac{1}{2}}{\frac{3}{4}} = \frac{2}{3}$

(c) We are tasked with finding: $P(Ph_C = O|Ph_X = A)$.
Zachary has type O blood, so his genotype is G_Z=OO. For their child to have type O blood, the child must inherit an O allele from both Xena and Zachary.
If Xena's genotype is $G_X = AO$, she can pass down an O allele with probability $\frac{1}{2}$.
If Xena's genotype is $G_X = AA$, she cannot pass down an O allele (probability 0).
Use the law of total probability:
$P(Ph_C = O|Ph_X = A) = P(Ph_C = O|G_X = AO)P(G_X = AO|Ph_X = A) + P(Ph_C = O|G_X = AA)P(G_X = AA|Ph_X = A)$
$P(Ph_C = O|Ph_X = AO) = \frac{1}{2}$(AO passes O) $\cdot$ 1(OO always passes O) $= \frac{1}{2}$
$P(G_X = AO|Ph_X = A) = \frac{2}{3}$(from part b).
$P(Ph_C = O|G_X = AA) = 0$($G_X = AA$ cannot pass O)
$P(G_X = AA|Ph_X = A) = \frac{1}{3}$(complement of $P(G_X = AO|Ph_X = A)$).
Thus, $P(Ph_C = O|Ph_X = A) = \frac{1}{2} \cdot \frac{2}{3} + 0 \cdot \frac{1}{3} = \frac{1}{3}$

(d) We are tasked with finding:$P(G_X = AO|Ph_C = A, Ph_X = A)$.
Use Bayes' Theorem:

$$P(G_X = AO|Ph_C = A, Ph_X = A) = \frac{P(Ph_C = A|G_X = A0, Ph_X = A)P(G_X = AO|Ph_X = A)}{P(Ph_C = A|Ph_X = A)}$$

$P(Ph_C = A|G_X = A0, Ph_X = A) = P(Ph_C = A|G_X = A0) = \frac{1}{2}$. Since Xena's genotype is AO, she passes an A allele with probability $\frac{1}{2}$, and Zachary always passes O, so the child will have type A blood with probability $\frac{1}{2}$.
$P(G_X = AO|Ph_X = A) = \frac{2}{3}$(from part b).
$P(Ph_C = A|Ph_X = A) = 1 - P(Ph_C = O|Ph_X = A)$(from part c) $= \frac{2}{3}$

$$P(G_X = AO|Ph_C = A, Ph_X = A) = \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{2}{3}} = \frac{1}{2}$$

4. Partitioning the Deck (12 points)

You have a non-standard deck of 5 suits of (Ace,2,3,4,5,6,7,8,9,10,Jack,Queen), for a total of 60 cards. You will deal the cards uniformly at random into 4 hands, each containing 15 of the cards (so each card ends up in exactly one hand). Let A_i be the event that hand i has exactly one of the five aces. In this problem, we'll calculate $P(A_1 \cap A_2 \cap A_3)$.

(a) We might hope that the A_i are independent of each other (it would make the calculation easier...). Prove that A_1 and A_2 are not independent by appropriate calculations.

(b) Calculate $P(A_1 \cap A_2 \cap A_3)$. You must use the chain rule!

Ans:

(a) A_1 and A_2 are not independent if $P(A_1 \cap A_2) \neq P(A) \cdot P(B)$.

The total number of ways to select 15 cards for one hand is: $\binom{60}{15}$
The number of ways to select 1 Ace for one hand out of the 5 Aces is: $\binom{5}{1}$
The remaining 14 cards in one hand select from the 55 non-Aces: $\binom{55}{14}$

$$P(A_1) = P(A_2) = \frac{\binom{5}{1} \cdot \binom{55}{14}}{\binom{60}{15}}$$

Chose the hand 1 then chose hand 2:

$$P(A_1 \cap A_2) = \frac{\binom{5}{1} \cdot \binom{55}{14}}{\binom{60}{15}} \cdot \frac{\binom{4}{1} \cdot \binom{41}{14}}{\binom{45}{15}}$$

$$P(A_1) \cdot P(A_2) = (\frac{\binom{5}{1} \cdot \binom{55}{14}}{\binom{60}{15}})^2 \neq P(A_1 \cap A_2)$$

Thus, A_1 and A_2 are not independent.

(b) A_1, A_2, and A_3 are not independent:

$$\begin{aligned} P(A_1 \cap A_2 \cap A_3) &= P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \text{[Chain Rule]} \\ &= P(A_1) \cdot \frac{P(A_1 \cap A_2)}{P(A_1)} \cdot P(A_3|A_1 \cap A_2) \text{[Bayes' Rule]} \\ &= P(A_1 \cap A_2) \cdot P(A_3|A_1 \cap A_2) \end{aligned}$$

$P(A_1 \cap A_2) = \frac{\binom{5}{1} \cdot \binom{55}{14}}{\binom{60}{15}} \cdot \frac{\binom{4}{1} \cdot \binom{41}{14}}{\binom{45}{15}}$(get from a).

Given that hands 1 and 2 already have 1 ace each, there are 3 remaining aces and 30 cards left to distribute among hands 3. The probability that hand 3 gets exactly 1 ace is: $P(A_3|A_1 \cap A_2) = \frac{\binom{3}{1}\cdot\binom{27}{14}}{\binom{30}{15}}$

Thus,

$$P(A_1 \cap A_2 \cap A_3) = \frac{\binom{5}{1} \cdot \binom{55}{14}}{\binom{60}{15}} \cdot \frac{\binom{4}{1} \cdot \binom{41}{14}}{\binom{45}{15}} \cdot \frac{\binom{3}{1} \cdot \binom{27}{14}}{\binom{30}{15}}$$

5. Secret Admirers (20 points)

Suppose you are using a dating app where you are presented with a list of n user profiles sequentially: you only get to traverse the list once, one user at a time. You do not get to look ahead or go back. For each user, you get two actions: match or pass. If you match, you can no longer look at other people's profiles and commit to your match. If you pass, you never get to see them again. The app guarantees that whoever you "match" with will agree to go out on a date with you. Your goal is to maximize your chances of finding the best date out the n people. You are torn: match too early, you miss out on someone better who may come along later. Match too late and you might let "the one" slip away.

You don't know much about your potential dating pool (so you can't estimate the chances that the current person is the best), but you do know what you're looking for — you can immediately rank a new profile relative to those you have already seen. You also know that the app will show you the n profiles in a uniformly random order.

A probability expert tells you that the optimal strategy is as follows: Reject the first q - 1 admirers you encounter (regardless of how good you think they are) for some number q. Starting with profile q, you will match with the first profile who is better than everyone you have seen so far.

In this problem, we'll compute the best value of q.

You may assume that $n \geq 1$.

(a) First, for a baseline, suppose your strategy were instead to match with the third profile no matter what. What is your probability of matching with your favorite among the n profiles?

(b) Now, let's start analyzing our strategy. For two natural numbers $q \leq i$, compute the probability that the best profile among the first q - 1 is also the best profile among the first i - 1 (so the max[1, i) = max[1, q)). You may assume $1 < q \leq i \leq n$.

(c) You match with the first profile at index q or later that is better than all the prior profiles you have seen. Supposing that the best profile is at index i, what is the probability that you will match with the best profile? Unlike in the previous part, for this part you will also need to handle the case that $i < q$; you may still assume that $1 < q$. (Hint: use part(b)!)

(d) We now set up a formula for the probability of selecting the best match if we ignore everyone before an arbitrary point q (i.e., we only start considering matching with someone if they are the q^{th} person we see or above). Use the Law of Total Probability to express the quantity as a summation over all possible placements of the best match. You will need to reason about the definition of our events to come up with the final result. Previous parts may be helpful here. The final answer is not "pretty" for this problem (ours still has a summation in it, for example); simplify as far as you can, but don't expect a clean final

answer. You also might need to have a separate formula for very small values of q or n (we have a special case when q = 1. If you have a separate case, you should explain where it comes from). To help you confirm if your answer is correct, when n = 10 and q = 5, the probability is approximately 0.3983, when n = 10 and q = 4 the probability approximately 0.3987.

(e) If n = 100, what is the best value of q? If n = 1000, what is the best value of q?
You do not need to provide an explanation for this part, but you may find it helpful to write a program or use graphing software for this part.

Ans:

(a) The probability of the third profile being the best out of n profiles is straightforward. Since the profiles are presented in a uniformly random order:

$$P(\text{third profile is best}) = \frac{1}{n}$$

(b) Each profile in [1, i - 1] is possible be the best profile. So the total sample space should be i - 1. The event that the best profile in [1, q - 1] remains the best in [1, i - 1] is when the best profile in [1, q - 1]. So,

$$P(Best[1, q-1] = Best[1, i-1]) = \frac{q-1}{i-1}$$

(c) If the best profile is at index i, the following cases occur:
1. $i < q$: You will never match because you reject all profiles before index q. Thus, $P(match|i < q) = 0$.
2. $i \geq q$: Since our strategy is match with the first profile who is better than everyone you have seen so far. So if i is the best profile, we can not chose the profile between [q and i - 1] in previous process. In other words, we need Best[1, q - 1] = Best[1, i - 1]. (so as same answer as the part b)

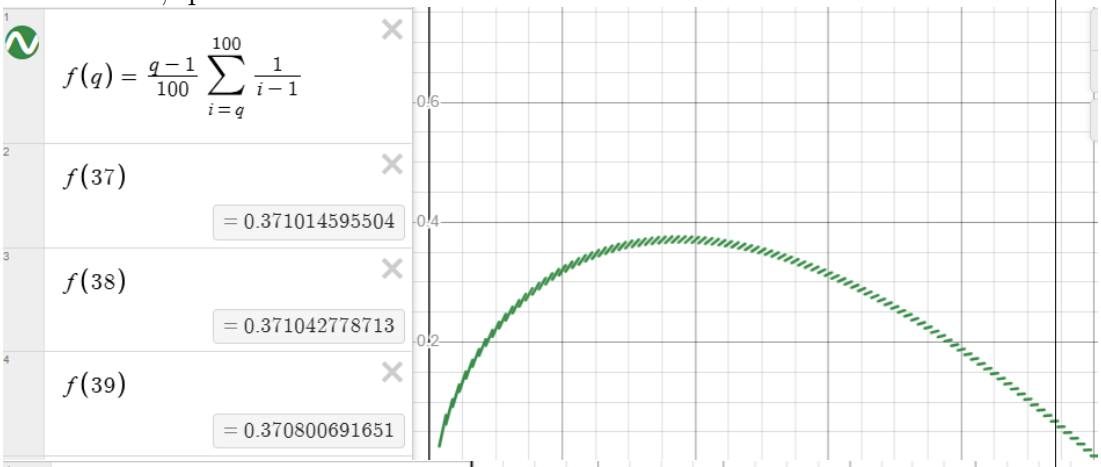
$$P(\text{match with best}|\text{best at i}) = \frac{q-1}{i-1}$$

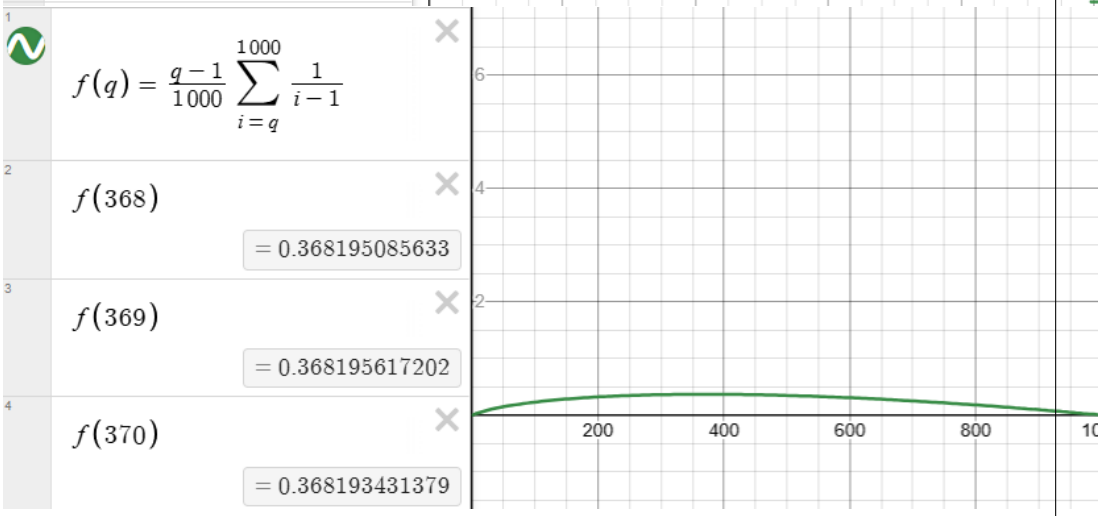
(d) Special Case q = 1:
When q = 1, you match with the first profile that appears, meaning you always match with the best profile if it happens to be the first one shown. Thus P(match with best) = P(best is at i = 1) = $\frac{1}{n}$

Using the Law of Total Probability, we sum over all possible positions i of the best profile:

$$\begin{aligned} P(\text{match with best}) &= \sum_{i=q}^{n} P(\text{Best at i}) \cdot P(\text{match with best}|\text{best at i}) \\ &= \sum_{i=q}^{n} \frac{1}{n} \cdot \frac{q-1}{i-1} \\ &= \frac{1}{n} \cdot (q-1) \sum_{i=q}^{n} \cdot \frac{1}{i-1} \end{aligned}$$

(e) If n = 100, q = 38 is the best value.
If n = 1000, q = 369 is the best value





6. Naive Bayes (Coding, 20 points)

   (a) Implement the function fit.

   (b) Implement the function predict.

Ans:

```
        def get_counts(filenames:List[str]) -> Dict[str, int]:
            counts = {}
            for filename in filenames:
                words = self.word_set(filename)  # Get unique
                    words in the email
                for word in words:
                    counts[word] = counts.get(word, 0) + 1 #
                        Count occurrences of each word
            return counts

        # Count ham and spam emails
        self.num_train_hams = len(train_hams)
        self.num_train_spams = len(train_spams)

        # Calculate word frequencies for ham and spam
        self.word_counts_ham = get_counts(train_hams)
        self.word_counts_spam = get_counts(train_spams)
```

```
        words = self.word_set(filename)
        # Log probabilities for ham and spam
        log_prob_ham = np.log(self.num_train_hams /
            (self.num_train_hams + self.num_train_spams))
        log_prob_spam = np.log(self.num_train_spams /
            (self.num_train_hams + self.num_train_spams))
        # Total number of words in ham and spam training sets
        # Compute log probabilities for the given email
        for word in words:
            ham_count = self.word_counts_ham.get(word, 0)
            spam_count = self.word_counts_spam.get(word, 0)
            # Apply Laplace smoothing
            log_prob_ham += np.log((ham_count + 1) /
                (self.num_train_hams + 2))
            log_prob_spam += np.log((spam_count + 1) /
                (self.num_train_spams + 2))
        # Return the label with the higher log probability
        return self.HAM_LABEL if log_prob_ham >= log_prob_spam
            else self.SPAM_LABEL
```

7. Feedback (1 points)

- I spent approximately 8 hours on this assignment.
- I spent the most time on Problem 5: Secret Admirers
- I create a code with more test accuracy than the "solution" expect (which I submit), but I get no point in accuracy. Theoretically, using total_ham_words and total_spam_words as denominators aligns better with the Naive Bayes assumption and may perform better when the lengths and word distributions of emails vary. This method captures word frequencies more effectively and provides more granular probabilities. However, the choice should ultimately depend on experimental results. If the accuracies of both methods are similar, the simpler approach (self.num_train_hams and self.num_train_spams) might be preferred for ease of implementation and faster computation.

```
words = self.word_set(filename)
# Log probabilities for ham and spam
log_prob_ham = np.log(self.num_train_hams / (self.num_train_hams + self.num_train_spams))
log_prob_spam = np.log(self.num_train_spams / (self.num_train_hams + self.num_train_spams))

# Total number of words in ham and spam training sets
# total_ham_words = sum(self.word_counts_ham.values())
# total_spam_words = sum(self.word_counts_spam.values())

# Compute log probabilities for the given email
for word in words:
    ham_count = self.word_counts_ham.get(word, 0)
    spam_count = self.word_counts_spam.get(word, 0)

    # Apply Laplace smoothing
    # log_prob_ham += np.log((ham_count + 1) / (total_ham_words + 2))
    # log_prob_spam += np.log((spam_count + 1) / (total_spam_words + 2))
    log_prob_ham += np.log((ham_count + 1) / (self.num_train_hams + 2))
    log_prob_spam += np.log((spam_count + 1) / (self.num_train_spams + 2))
# Return the label with the higher log probability
return self.HAM_LABEL if log_prob_ham >= log_prob_spam else self.SPAM_LABEL
```

- It is a fun coding program.