CS 344 (Spring 2017): End-semester Examination*

Instructor: Shivaram Kalyanakrishnan

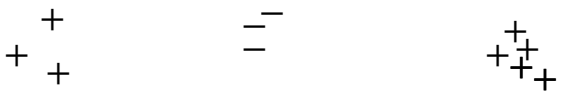
2.00 p.m. – 5.00 p.m., April 21, 2017, 101/103/105 New CSE Building

Total marks: 35

Note Provide brief justifications and/or calculations along with each answer to illustrate how you arrived at the answer.

Question 1. This question pertains to ensemble methods for binary classification. An ensemble comprises $k \geq 2$ base classifiers $L_1, L_2, \ldots, L_k$. The prediction made by the ensemble is the (unweighted) majority of the base classifiers' individual predictions. Assume ties are broken uniformly at random.

1a. Consider the data set shown below: 11 points in 2-dimensional space, each labeled "+" or "-". If the base classifier used is a Perceptron (a line in the 2-dimensional space, partitioning it into a "+" region and a "-" region), what is the minimum number of base classifiers needed to classify all the input points correctly? You don't have to give a proof. Draw the figure below and draw each of the Perceptrons in the minimal ensemble you construct. [2 marks]



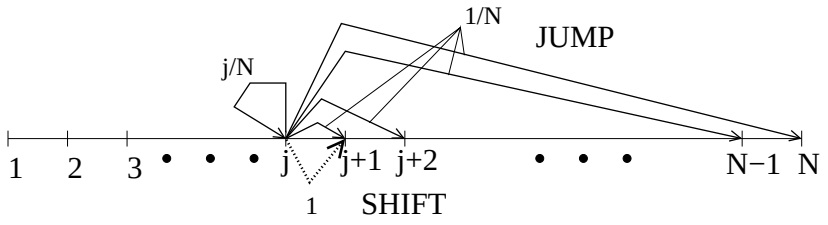
1b. *Bagging* is an ensemble method, which trains its base classifiers as follows. Given a data set D, bagging creates k derived data sets $D_1, D_2, \ldots, D_k$, where for each $j \in \{1, 2, \ldots, k\}$, each row of D_j (of the form $(\boldsymbol{x}, y)$) is drawn uniformly at random from among the rows of D (hence there could be repeated rows). It is typical to make all the derived data sets the same size. Subsequently, base classifier L_j is obtained by training independently (applying the base learning method—SVM, Perceptron, decision tree, etc.) on data set D_j, for $j \in \{1, 2, \ldots, k\}$. Compare bagging with boosting. List at least one advantage each method has over the other. [2 marks]

1c. The reason ensemble methods perform well is because the errors of the base classifiers tend not to be correlated. Suppose a randomised procedure to generate a base classifier guarantees that the base classifier will misclassify at most 10% of the points in a data set, and further, the points it misclassifies will be drawn uniformly at random from the data set. If the procedure is applied to produce $k = 5$ base classifiers, show that the probability that the (unweighted majority-voted) 5-member ensemble correctly classifies a point is at least 99%. [2 marks]

*Question 4 is based on an exercise in the textbook by Russell and Norvig (2010).

Question 2. You are placed on the number line shown below. You can occupy any of the positions in $\{1, 2, \ldots, N\}$, where $N \geq 2$. Your objective is to get to position N "quickly".

You have nothing more to do—your task is over—if you have reached position N. If, however, you are in some position $j \in \{1, 2, \ldots, N-1\}$, you can choose to perform one of two actions: SHIFT and JUMP. The SHIFT action is deterministic: it takes you to position $j+1$. JUMP, however, has a stochastic outcome. From position j, JUMP has a probability of $1/N$ of taking you to any fixed position ahead of j (of which there are $N - j$). With the remaining probability of j/N, JUMP keeps you at position j itself. The figure below shows all possible transitions from position j; each transition is annotated with the corresponding probability.



In this exercise, you will work out an *optimal policy* for you to follow. A policy here is a mapping from $\{1, 2, \ldots, N-1\}$ to $\{\text{SHIFT}, \text{JUMP}\}$. An optimal policy *minimises* the expected number of actions taken to reach position N; it does so for every starting position $j \in \{1, 2, \ldots, N-1\}$. Let $\pi^\star$ be an optimal policy, and for $j \in \{1, 2, \ldots, N-1\}$, let $T^\star(j)$ denote the expected number of actions that are performed, starting from position j and following $\pi^\star$, in order to reach position N.

2a. Show that $\pi^\star(N-1) = \text{SHIFT}$. What is $T^\star(N-1)$? [1 mark]

2b. For $j \in \{2, 3, \ldots, N-1\}$, show that if $\pi^\star(j) = \text{JUMP}$, then $\pi^\star(j-1) = \text{JUMP}$. [3 marks]

2c. From 2a and 2b, it follows that there is an optimal policy that selects JUMP for positions $\{1, 2, \ldots, k\}$, and SHIFT for positions $\{k+1, k+2, \ldots, N-1\}$, for some particular $k \in \{1, 2, \ldots, N-1\}$. Show that $k = N - \Theta(\sqrt{N})$. [4 marks]

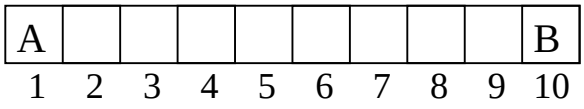
2d. Plot $T^\star(j)$ as a function of $j \in \{1, 2, \ldots, N-1\}$. [2 marks]

Question 3. This question is about conditional probabilities.

3a. There are three boxes, each containing two coins. One box contains two gold coins, one box contains two silver coins, and the other box contains a gold coin and a silver coin. A box is chosen uniformly at random from among the three boxes, and a coin is selected uniformly at random from among the coins in the chosen box. If the selected coin is a gold coin, what is the probability that the other coin in the chosen box is also a gold coin? [2 marks]

3b. Consider Boolean random variables A, B, and C. A takes values a and $\bar{a}$, B takes values b and $\bar{b}$, and C takes values c and $\bar{c}$. If it is known that $P(a|c) \geq P(b|c)$ and $P(a|\bar{c}) \geq P(b|\bar{c})$, can we conclude that $P(a) \geq P(b)$? Provide a proof or a counterexample. [2 marks]

Question 4. The figure below depicts a zero sum game played by agents A and B. The game is played on a board of length $n \geq 3$, which comprises cells $1, 2, \ldots, n$ in sequence. The figure is drawn for $n = 10$.



The game begins with A on cell 1, and B on cell n (as shown in the figure). The players take turns to make moves, with A making the first move. To make a move, an agent can either (1) shift to an empty cell beside its current cell (if one is available), or (2) if the other agent is in a cell next to it, "jump" over that agent to the empty cell next to it (if one is present). Let (x, y) represent a state in which A is on cell x and B is on cell y. As per the rules of the game, if A has to make a move from $(4, 6)$, it can go to either $(3, 6)$ or $(5, 6)$. If A has to make a move from $(1, 7)$, it can only go to $(2, 7)$. If B has to make a move from $(4, 5)$, it can go to either $(4, 3)$ or $(4, 6)$.

If any one of the agents reaches the opposite end, the game ends and that agent is declared the winner. In other words, if a state (n, x) is reached, then A wins. If $(x, 1)$ is reached, then B wins. If, after a total of $2n$ moves, no player has reached the opposite end, then the game ends in a draw.

4a. For the game with $n = 10$, will A ever have to make a move starting from $(4, 6)$? If yes, describe a sequence of moves from the starting configuration (recall that A moves first) that leads to this situation. If not, prove that there exists no such sequence. [2 marks]

4b. For each $n \geq 3$, describe the outcome of a game in which A and B both play minimax strategies. Prove that your answer is correct. (Hint: You will find it useful to work out $n = 3$ and $n = 4$ as special cases, and then generalise your results.) [4 marks]

Question 5. Define the *Markov Blanket* of a node X in a Bayes Net in terms of the *Parent* and *Child* relations, and necessary logical operators. How does the Markov Blanket find use in Gibbs Sampling? [2 marks]

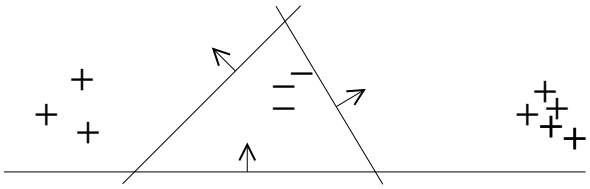
Question 6. Take the Traveling Salesperson Problem, which can be equivalently stated in graph-theoretic terms: Given a complete weighted graph, find a Hamiltonian cycle (a cycle that visits every vertex exactly once) with the least weight. Since the problem is NP-hard, in practice we aim to find a Hamiltonian cycle with as low weight as possible in a reasonable amount of time.

How would you model and solve the Traveling Salesperson Problem using Hill Climbing? Comment on the design choices you have to make, and the effects of these choices on the quality of the solution found and the time taken to find it. [3 marks]

Question 7. What are n-gram character and word models? What is the tradeoff to be balanced in identifying a suitable value of n in these models? How are n-gram models applied to spam detection? [4 marks]

Solutions

1a. At least three Perceptrons are needed in order to classify all the data points correctly. Below is one possible arrangement: observe that each data point has some two Perceptrons that classify it correctly, and one that does not.



1b. Boosting is inherently *sequential*: the $(j+1)$-st classifier is trained based on the errors made by the first j classifiers on the training data. In bagging, on the other hand, each classifier is learned independently. As a consequence of this difference, bagging affords much more flexibility in implementation. Specifically, the base classifiers can be trained *in parallel*, and the training process can be much quicker than that of boosting. However, observe that the errors of the base classifier will only be *uncorrelated* under bagging, whereas in boosting, each classifier's error is expressly *anti-correlated* with the aggregate error of the classifiers preceding it. Hence, for the same ensemble size, boosting is capable of yielding a more accurate ensemble.

1c. A point will be misclassified by the ensemble only if a majority of classifiers (that is, exactly 5, 4, or 3 classifiers) misclassify the point. Since each classifier independently has at most a probability of $\epsilon = 10\%$ of misclassifying the point, the probability of misclassification by the ensemble is upper-bounded by

$$\binom{5}{5}\epsilon^5(1-\epsilon)^0 + \binom{5}{4}\epsilon^4(1-\epsilon)^1 + \binom{5}{3}\epsilon^3(1-\epsilon)^2 = 10^{-5} + 5\times 10^{-4}\times 0.9 + 10\times 10^{-3}\times 0.81 = 0.00865.$$

Thus, the accuracy of the ensemble is at least $1 - 0.00865 > 99\%$.

2a. For $j \in \{1, 2, \ldots, N-1\}$, denote by $T^\star(j, \text{SHIFT})$ the expected number of steps to reach position N by taking the SHIFT action from position j once, and thereafter acting optimally (according to $\pi^\star$). Similarly, denote by $T^\star(j, \text{JUMP})$ the expected number of steps to reach position N by taking the JUMP action from position j once, and thereafter acting according to $\pi^\star$. $T^\star(\cdot,\cdot)$ is analogous to the action value function $Q^\star(\cdot,\cdot)$ that we studied in class. However, note that we wish to *minimise* the expected number of steps to reach N (Q is conventionally maximised). Thus,

$$T^\star(j) = \min(T^\star(j, \text{SHIFT}), T^\star(j, \text{JUMP})).$$

From the transition probabilities specified, we obtain:

$$\begin{aligned} T^\star(N-1, \text{SHIFT}) &= 1, \text{ and} \\ T^\star(N-1, \text{JUMP}) &= 1 + \frac{N-1}{N} T^\star(N-1) > 1. \end{aligned}$$

Therefore, $\pi^\star(N-1) = \text{SHIFT}$ and $T^\star(N-1) = 1$.

2b. We are given $T^\star(j, \text{JUMP}) \leq T^\star(j, \text{SHIFT})$, and we have to show that $T^\star(j-1, \text{JUMP}) < T^\star(j-1, \text{SHIFT})$. Clearly, $T^\star(j-1, \text{SHIFT}) = 1 + T^\star(j, \text{JUMP})$. Therefore, we need to show that $T^\star(j-1, \text{JUMP}) < 1 + T^\star(j, \text{JUMP})$. We will prove a stronger result, that

$$T^\star(j-1, \text{JUMP}) = T^\star(j, \text{JUMP}).$$

$$\begin{aligned} T^\star(j-1, \text{JUMP}) &= 1 + \frac{j-1}{N} T^\star(j-1) + \frac{1}{N} \sum_{r=j}^{N-1} T^\star(r) \\ &\leq 1 + \frac{j-1}{N} T^\star(j-1, \text{JUMP}) + \frac{1}{N} \sum_{r=j}^{N-1} T^\star(r) \\ &\Longrightarrow \\ T^\star(j-1, \text{JUMP}) &\leq \frac{1}{N-j+1} \left(N + \sum_{r=j}^{N-1} T^\star(r) \right). \end{aligned}$$

Now, since $T^\star(j) = T^\star(j, \text{JUMP}) = 1 + \frac{j}{N} T^\star(j, \text{JUMP}) + \frac{1}{N} \sum_{r=j+1}^{N-1} T^\star(r)$, we get

$$T^\star(j) = T^\star(j, \text{JUMP}) = \frac{1}{N-j} \left(N + \sum_{r=j+1}^{N-1} T^\star(r) \right).$$

Therefore,

$$\begin{aligned} T^\star(j-1, \text{JUMP}) &\leq \frac{1}{N-j+1} \left(N + \sum_{r=j}^{N-1} T^\star(r) \right) \\ &= \frac{1}{N-j+1} \left(N + T^\star(j, \text{JUMP}) + \sum_{r=j+1}^{N-1} T^\star(r) \right) \\ &= \frac{1}{N-j+1} \left(N + T^\star(j, \text{JUMP}) + (N-j) T^\star(j) - N \right) \\ &= T^\star(j, \text{JUMP}). \end{aligned}$$

Since $T^\star(j-1, \text{JUMP}) \leq T^\star(j, \text{JUMP})$ and $T^\star(j, \text{SHIFT}) > T^\star(j, \text{JUMP})$, it must be the case that $\pi^\star(j-1) = \text{JUMP}$. The implication is that the "$\leq$" in our first derivation is indeed an "$=$" (that is, $T^\star(j-1) = T^\star(j-1, \text{JUMP})$), and so $T^\star(j-1, \text{JUMP}) = T^\star(j, \text{JUMP})$.

2c. Let $k \in \{1, 2, \ldots, N-1\}$ be such that JUMP is an optimal action for positions $1, 2, \ldots, k$, and SHIFT is an optimal action for $k+1, k+2, \ldots, N-1$. This pattern implies that $T^\star(k, JUMP) \leq N - k$, which, expanded, becomes

$$\frac{1}{N-k} \left(N + \sum_{r=k+1}^{N-1} T^\star(r) \right) \leq N - k, \text{ or equivalently,}$$

$$\frac{1}{N-k} \left(N + \sum_{r=k+1}^{N-1} (N-r) \right) \leq N - k.$$

Simplifying this expression yields

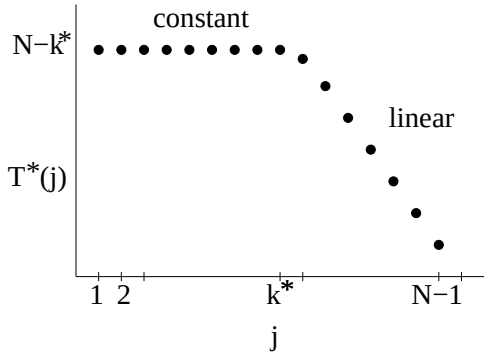
$$\left(N + \frac{1}{2} - k\right)^2 \geq 2N + \frac{1}{4}.$$

Since $k \leq N - 1$, we get

$$k \leq k^\star \stackrel{\text{def}}{=} \left\lfloor N + \frac{1}{2} - \sqrt{2N + \frac{1}{4}} \right\rfloor.$$

Given the result from 2b (in an optimal policy, every action preceding a JUMP is also a JUMP), we conclude that an optimal policy will pick JUMP at positions $1, 2, \ldots, k^\star$, and SHIFT at positions $k^\star + 1, k^\star + 2, \ldots, N - 1$, where $k^\star = N - \Theta(\sqrt{N})$.

2d. The plot below shows the dependence of $T^\star(j)$ on j. The function is constant up to $k^\star$: the actual value is in the interval $[N - k^\star - 1, N - k^\star]$. The function value at $k^\star + 1$ is exactly $N - k^\star - 1$; thereafter the function decreases by 1 with every increment to j.



3a. This problem is due to Bertrand.[1] Let the boxes be named B_1, B_2, and B_3. Box B_1 contains coins G_1 and G_2, box B_2 contains coins S_1 and S_2, and box B_3 contains coins G_3 and S_3. Using the fact that the occurrence of one coin excludes all others, we get

$$\begin{aligned}
P(B_1 | G_1 \text{ or } G_2 \text{ or } G_3) &= P(G_1 \text{ or } G_2 | G_1 \text{ or } G_2 \text{ or } G_3) \\
&= \frac{P((G_1 \text{ or } G_2) \text{ and } (G_1 \text{ or } G_2 \text{ or } G_3))}{P(G_1 \text{ or } G_2 \text{ or } G_3)} \\
&= \frac{P(G_1 \text{ or } G_2)}{P(G_1 \text{ or } G_2 \text{ or } G_3)} \\
&= \frac{\frac{1}{3}\frac{1}{2} + \frac{1}{3}\frac{1}{2}}{\frac{1}{3}\frac{1}{2} + \frac{1}{3}\frac{1}{2} + \frac{1}{3}\frac{1}{2}} \\
&= \frac{2}{3}.
\end{aligned}$$

[1] See `http://blog.zymergi.com/2013/06/bertrands-box-paradox.html`.

3b.

$$
\begin{aligned}
& P(a|c) \geq P(b|c) \text{ and } P(a|\bar{c}) \geq P(b|\bar{c}) \\
& \iff P(a \text{ and } c)/P(c) \geq P(b \text{ and } c)/P(c) \text{ and } P(a \text{ and } \bar{c})/P(\bar{c}) \geq P(b \text{ and } \bar{c})/P(\bar{c}) \\
& \iff P(a \text{ and } c) \geq P(b \text{ and } c) \text{ and } P(a \text{ and } \bar{c}) \geq P(b \text{ and } \bar{c}) \\
& \implies P(a \text{ and } c) + P(a \text{ and } \bar{c}) \geq P(b \text{ and } c) + P(b \text{ and } \bar{c}) \\
& \iff P(a) \geq P(b).
\end{aligned}
$$

4a. Let x_A denote the cell occupied by A, and x_B the cell occupied by B. Let the *parity* of a state (x_A, x_B) denote $(x_A + x_b) \mod 2$. Observe that the parity flips if either agent moves to an adjoining cell, and the parity remains the same if an agent jumps over the other.

We are interested in the reachability of the state $(4, 6)$, with it being A's turn to move. Observe that A is to the left of B in this state: the state is therefore reachable only if the number of jump moves preceding it is even. Since it is A's turn, the total number of moves made before reaching it must also be even. Therefore, the number of moves that took either agent to an adjacent cell must also be even, which means the the parity must be the same as that of the start state. However, the parity of $(1, 10)$ is 1, while that of $(4, 6)$ is 0. We may conclude that $(4, 6)$ cannot be reached in an even number of moves.

4b. In the $n = 3$ game, the only possible sequence of moves is $(1, 3) \xrightarrow{A} (2, 3) \xrightarrow{B} (2, 1)$, which results in a win for B. In the $n = 4$ game, the first two moves are determined: $(1, 4) \xrightarrow{A} (2, 4) \xrightarrow{B} (2, 3)$. Hereafter, A can move either to $(1, 3)$ or to $(4, 3)$. The latter results in a win for A, and so A can force a win when $n = 4$. Upon closer inspection, we find that B can force a win if n is odd, and A can force a win if n is even.

Towards a proof, let us first consider odd n. B's winning strategy is as follows.

O1. If possible, move to the cell on the left; if not

O2. If possible, jump over A to a cell on its left; if not

O3. From the state $(1, 2)$, force the sequence $(1, 2) \xrightarrow{B} (1, 3) \xrightarrow{A} (2, 3) \xrightarrow{B} (2, 1)$.

It is not hard to see that if the first two options are not available to B, then the third one must be—and third option clearly leads to a win for B. In fact, observe that B's strategy must necessarily take it to a state of the form $(j, j + 1)$. Based on an argument similar to that in 4a, we observe that in such a state (which has a parity 1), it must be B's turn to move. $j = 1$ corresponds to O3 (B can force a win): if not, B will cross over A to reach $(j, j - 1)$. A can choose to cross back to $(j - 2, j - 1)$, or proceed right to $(j + 1, j - 1)$. Regardless of A's strategy, B will move at least one cell to the left with each move, and it will move two cells to the left at least once. Hence, after at most $n - 2$ moves, B will either reach the opposite end, or reach $(1, 2)$—both resulting in a win. The only possible way for A to deny B is by itself reaching its opposite end before B wins. However, A only moves 1 cell at a time, and because of B's strategy, it can never jump over B to the right. Hence, A needs at least $n - 1$ moves to reach the opposite end.

The argument for even n proceeds similarly, with A moving right if at all possible, otherwise arriving at $(n - 1, n)$, from which it can force a win. The key difference from odd-n games is that when a state of the form $(j, j + 1)$ is reached in an even-n game, it must be A's turn to move.

5. The Markov Blanket of X is defined as

$\{Y : (Y \neq X)$ and $(Y = Parent(X)$ or $Y = Child(X)$ or $(\exists Z : Z = Child(X)$ and $Y = Parent(Z)))\}$.

Let $X_1, X_2, \ldots, X_n$ be the nodes in the Bayes Net. Each step of Gibbs Sampling samples a node X_i, $i \in \{1, 2, \ldots, n\}$ (whose value is not given) with probability

$$P(X_i|X_1, X_2, \ldots, X_{i-1}, X_{i+1}, X_{i+2}, \ldots, X_n).$$

From the semantics of Bayes Nets, this probability is exactly equal to $P(X_i|\text{Markov Blanket of } X_i)$. The latter probability is less expensive to compute if if the size of the Markov Blanket of X_i is significantly smaller than n, which is often the case in practice.

6. The first aspect to consider when applying Hill Climbing to a problem is how solutions will be represented. For the Traveling Salesperson Problem, solutions are cycles of length n, where n is the number of vertices. Cycles can naturally be represented as strings (such as 14837256). To avoid redundancy (multiple strings mapping to the same cycle), it can be enforced that the first letter of the string is always 1, and the second letter is smaller than the last letter. Note that the number of unique cycles in a complete n-vertex graph is $(n-1)!/2$.

The natural value to associate with each cycle, in the case of the Traveling Salesperson Problem, would be the weight of the cycle.

The most important design choice is in fixing the set of neighbours for each cycle. If the neighbourhood is fixed, Hill Climbing proceeds by repeatedly evaluating all the neighbours of a particular cycle. If a neighbour has a lower weight than the cycle currently considered, then that neighbour becomes the "current" cycle (and thereafter its neighbours evaluated). This procedure is repeated until every cycle in the neighbourhood of the current cycle C is no lower in weight. If so, C is returned as the solution.

A common definition of neighbourhood is as follows: cycles C_1 and C_2 are k-neighbours if some k edges can be removed from C_1, and some k edges can be then added in order to obtain C_2. Thus, for example, 123456 and 124356 are 2-neighbours (remove 23 and 45 from 123456, and add 24 and 35 to obtain 124356). Naturally, other notions of locality can also be used to define neighbours. The main effect of the choice of neighbours is in the time taken to evaluate all the neighbours, as well as the quality of the solution eventually found. For example, the number of 2-neighbours of a cycle is smaller than the number of 3-neighbours: on average, we can expect the use of 2-neighbourhood for hill-climbing to find solutions more quickly than by using 3-neighbourhood. On the other hand, using 3-neighbourhood is likely to find better cycles at convergence.

7. n-gram models are language models: they map a given sequence of characters (or words) to a probability. The parameter n encodes an assumption on the degree of independence between characters in the string, based on their separation. Specifically, the probability

$$P(c_1 c_2 \ldots c_k) = \Pi_{i=1}^{k} P(c_i|c_1 c_2 \ldots c_{i-1})$$

is approximated in an n-gram model by

$$\Pi_{i=1}^{k} P(c_i|c_{i-n} c_{i-n+1} \ldots c_{i-1}).$$

Clearly, the larger the value of n, the more accurate the model will be—as long as every probability in the calculation is exact. However, the main use of n-gram models is *in practice*, wherein $P(c_i|c_{i-n}c_{i-n+1}\dots c_{i-1})$ are estimated based on frequencies occurring in a particular data set. When this is done, unfortunately there is less data available for larger values of n, and so probability estimates can be noisy. Hence, it is common to not go beyond $n = 3$ or $n = 4$, and to *fall back* on $n = 1$ and $n = 2$ when data is insufficient for predicting with larger values of n (for example, some sequences can have even *zero* counts in the data set).

n-gram models can be used for spam detection by estimating

$$P(\text{spam}|c_1c_2\dots c_k) \propto P(\text{spam})P(c_1c_2\dots c_k|\text{spam}) \approx P(\text{spam})\Pi_{i=1}^{k}P(c_i|c_{i-n}c_{i-n+1}\dots c_{i-1}, \text{spam}).$$

Here, $P(c_i|c_{i-n}c_{i-n+1}\dots c_{i-1}, \text{spam})$ is computed based on frequencies in a spam data set. A similar model can be learned for non-spam data sets, and used to estimate $P(\text{non-spam}|c_1c_2\dots c_k)$. The prediction of spam/non-spam can be made based on these probabilities.