1) **Nearest Neighbor Classifier**
   a) Under what conditions would you expect the kNN (k-Nearest Neighbor) classifier to give better results than the NN (Nearest Neighbor) classifier?
   
b) How are the prior probabilities $P(\omega_i)$ reflected in the NN and kNN classification methods? For example, consider character recognition: in English, the letter $e$ is much more frequent than the letter $q$. A Bayes’ classifier is affected by this, because it uses the value of $P(\omega_i)$ in the process of making a classification decision. In contrast, an NN classifier for OCR does not make explicit reference to $P(\omega_i)$; it just looks for the nearest neighbor. So how do the prior probabilities have an effect on an NN classifier?

2) **Estimation of discriminant functions** Describe the steps used to design a classifier based on estimation of discriminant functions (Ch 5 of DHS, pp. 93-95 of the course reader). A brief, general description of the design process is enough. No equations are needed. Provide enough information to answer the following questions:
   - How many discriminant functions have to be estimated?
   - How are the design samples used?

3) **Decision trees** Here is a decision tree for a two-class problem with three binary features ($x_1, x_2, x_3$). Class $\omega_1$ is defined by the Boolean formula “($x_1$ AND $x_2$) OR $x_3$”. This means that for samples in class $\omega_1$, either both features $x_1$ and $x_2$ are present, or feature $x_3$ is present. Class $\omega_2$ is the opposite.

![Decision Tree Diagram](image)

   a) Find the expected length from the root to a leaf for this tree. Assume that the values 0 and 1 are equally likely for each of the three features $x_1, x_2, x_3$. This means that each of the 8 feature vectors (000, 001, 010, etc) is equally likely, and as a result, some leaves are reached more often than other leaves.

   b) Show an alternate decision tree that performs the same classification, but has shorter average path length. State the average path length of your tree.

4) **Multiple classifiers** Classifiers A, B, and C each perform digit recognition (10 classes). Each classifier is correct 50% of the time. Assume that the classifiers are independent: if classifier A makes an error on a certain input, that does not affect the probability that classifier B makes an error on this input. Also, if classifiers A and B both make errors on an input, there is no correlation among the wrong answers given by A and B. (Given that A and B both made an error, the chance that they both gave the same wrong answer is 1/9.)

   Classifier D is defined as a combination of classifiers A, B, C. It operates as follows:
   - Apply classifiers A, B, C to the input
   - If two or three of A, B, C produce the same answer $\omega_i$, then classifier D answers $\omega_i$
   - If all three of A, B, C produce different answers, then classifier D rejects the input.

   (a) What is $P$(correct) for classifier D? What is $P$(reject)? What is $P$(error)? (See discussion below.)

   (b) Now assume that classifiers A, B, and C are each correct 70% of the time, and find $P$(correct), $P$(reject), $P$(error).

   Performance of Classifier D can be improved in two ways:
   - improve the component classifiers, as illustrated by the improvement from 50% to 70% correct in (a) and (b).
   - use more classifiers. For example, using 5 classifiers, guess $\omega_i$ if three or more classifiers produce answer $\omega_i$. Reject if no $\omega_i$ appears more than twice in the list of guesses produced by the 5 classifiers.

   Optional part (c): estimate the correct classification rate for majority-voting combination of 5 classifiers, where each classifier is correct 70% of the time. Compare this to (b), to note the improvement in going from 3 to 5 classifiers.
Discussion for problem 4  There are many ways to calculate the answers. You need to find the probability of the various situations that cause the classifier to be right, wrong, or reject. Since \( P(\text{correct}) + P(\text{reject}) + P(\text{error}) = 1 \), you only need to figure out two of these three quantities. Here is a summary of the situations:

Classifier D rejects if
- A, B, C produce three different answers. All three might be wrong answers, or one of the answers might be right.

Classifier D is correct if
- all three of ABC give the right answer.
- two of ABC are right (3 ways this can happen: AB right, C wrong; BC right, A wrong; AC right, B wrong).

Classifier D is wrong if
- all three of ABC give the wrong answer AND at least two of these wrong answers agree.
- two of ABC give the same wrong answer, and the third classifier gives the right answer.

Introductory Problems about Grammars

These problems should not take long. Ask for help if you get stuck (particularly if you have never studied grammars before).

5) Here is a context-free grammar:

\[
\begin{align*}
\text{<start>} & \rightarrow \text{<letter> <start>} \\
\text{<start>} & \rightarrow \text{<letter>} \\
\text{<letter>} & \rightarrow \text{<vowel> | <cons>} : \text{The “|” symbol stands for “or”} \\
\text{<vowel>} & \rightarrow \text{a | e | i | o | u | y} \\
\text{<cons>} & \rightarrow \text{b | c | d | f | g | h | j | k | l | m} | \text{n} | \text{p} | \text{q} | \text{r} | \text{s} | \text{t} | \text{v} | \text{w} | \text{x} | \text{y} | \text{z}
\end{align*}
\]

a) Show a parse tree for the string “eat”.

b) This grammar is ambiguous. Find an example of a string that can be parsed in two ways.

c) Modify this grammar so that a letter “q” must be followed by a letter “u”.

d) Modify this grammar so that it can parse a string if and only if that string contains at least one vowel.

6) Page 176 of the course reader shows a context sensitive grammar for the square language \( a^nb^n c^n d^n \). Alter this grammar so that it instead produces the rectangle language \( a^nb^m c^n d^m \).

7) This problem gives you an idea of how general computations can be performed with a type-0 grammar. However, I am not recommending that you adopt type-0 grammars as your favourite programming language!

The following type-0 grammar computes powers of two: it generates the language \( \{a^i | i \text{ is a power of 2} \} \). Examples of strings it generates are \( a^1, a^2, a^4, a^8 \); this is shorthand for \( a, aa, aaaa, aaaaaaaa \).

\[
\begin{align*}
0 & : S \rightarrow a \\
1 & : S \rightarrow ACaB \\
2 & : Ca \rightarrow aaC \\
3 & : CB \rightarrow DB \\
4 & : CB \rightarrow E \\
5 & : aD \rightarrow Da \\
6 & : AD \rightarrow AC \\
7 & : aE \rightarrow Ea \\
8 & : AE \rightarrow \varepsilon
\end{align*}
\]

(a) Show a derivation of the string aaaaaaaa, a string of \( 2^3 \) a's. An ordered list of production-numbers is sufficient.

(b) Write a type-0 grammar that generates \( \{a^i | i = 2^j + 3^j \text{ for some positive integer } j \} \). This language includes aaaaa (j=1), aaaaaaaaaaa (j=2), and \( a^{35} \) (j=3).