Due date: February 18, 2025

This Problem Set consists of 4 questions, but you are free to skip *one* between {Problem 1, Problem 2}. The final deliverable should consist of solutions for 3 problems total. Submitting all 4 will not result in extra credit, though they make up nice exercises. :)

Problem 1: Distance & Metrics. Which of the following distance functions covered in the class are metrics: (a) Minkowski distance with respect to a convex set, (b) cosine distance, (c) edit distance, (d) dynamic time warping.

Justify your answer, namely, if it is a metric then argue that the three properties hold, and if it is not a metric then give a counter-example.

Problem 2: L_p-norms in $\mathbb{R}^n$. In class we learned about the p-norm of $x \in \mathbb{R}^n$, defined by

$$\|x\|_p = \begin{cases} (\sum_{i=1}^n |x_i|^p)^{1/p} & p \geqslant 1 \\ \max_i |x_i| & p = \infty. \end{cases}$$

as well as the induced metric distance $d_p(x, y) = \|x - y\|_p$.

(1) An interesting fact about p-norms is what is known as the "L_p inclusion:" for $1 \leqslant p < q \leqslant \infty$ [note q can be ∞], we have $\|x\|_p > \|x\|_q$. Prove this claim using the following:

- First show that if $\|x\|_p = 1$, then $\|x\|_q < 1$. **Hint**: *raise to appropriate powers*.
- Complete the proof for general x by relating it to $x/\|x\|_p$.

(2) Given a norm $\|\cdot\|$, a unit circle is defined by $B = \{x \in \mathbb{R}^n : \|x\| = 1\}$. Sketch the unit balls in $\mathbb{R}^2$ under p-norm for each $p \in \{1, 2, 5, 10, \infty\}$. What do you observe? How do these observations connect to (1)'s results? (If you are using LaTeX, consider pasting a figure or using `tikz`.)

(3) For $p \in (0, 1)$, we also define $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. Sketch the "unit ball" under this "metric." You may notice that this shape looks weird compared to the ones in (2). This is because the "unit ball" for $p \in (0, 1)$ is not *convex*. Prove that $d(x, y) = (\sum_{i=1}^n |x_i - y_i|^p)^{1/p}$ for $p \in (0, 1)$ does not qualify as a metric, hence the quotation marks.

Problem 3: Polynomial-time 1D k-means. Let $X = \{x_1, \ldots, x_n\}$ be a set of real values (i.e., a set of n points in $\mathbb{R}^1$). Describe a polynomial-time algorithm for computing an optimal k-means clustering of X. (**Hint:** *Sort the input values and use dynamic programming*.)

Problem 4: Huffman Coding Implementation. This problem asks you to implement Huffman coding. Your final deliverable consists of a write-up (see below for details) as well as the source code. You may choose any programming language you prefer, as this problem will be graded mostly on the empirical results (though we will briefly check the code).

Please implement the following four modules. You may choose any function signature and return type.

- **ComputeFrequency**. Takes a text corpus as input and computes the frequency distribution of each character.
- **BuildHuffmanTree**. Takes the character frequency distribution as input and converts it into a Huffman encoding tree. You may find it helpful to implement a Huffman class in order to organize your code.
- **Encode**. Takes a Huffman encoding tree and a string message consisting of existing characters in the Huffman encoding tree as inputs. Outputs a binary string that uniquely represents this message according to the Huffman encoding tree.
- **Decode**. Takes a Huffman encoding tree and a binary string as inputs. Outputs a message as a string of existing characters. This decoding should be unique.

You are provided the short novel *The Strange Case Of Dr. Jekyll And Mr. Hyde* by Scottish author Robert Louis Stevenson as a data set. A plain text version is provided under `Modules/Assignments/CS390_HW2_JH.txt`. Your write-up should include the following results on the above data set.

(1) Compute the frequency distribution for all characters that appear in the novel and plot the distribution. Report the the number of distinct characters in the novel, and the number of nodes required by your Huffman tree.

(2) Build the Huffman tree for the novel. Plot the distribution of codeword lengths. Report the number of nodes in the first three layers of the tree.

(3) Calculate the average number of bits per character in your Huffman encoding. Compare this against the theoretical entropy of the character distribution.

(4) Encode the entire novel, then decode it. Show a few example outputs and show that the decoded text matches the original input. For example, report a few sentences. For each sentence, print the original sentence, then print its Huffman code, and finally print the decoded output. (If done correctly, after decoding, you should re-obtain an exact copy of the original file.)

(5) Measure the runtime of each module. If you identify a bottleneck, what are some potential improvements?