# COMPSCI 589: Machine Learning - Homework 1

Vishnu Vardhan Reddy Bheem Reddy

September 17, 2025

## Question 1: KNN and Data Scaling

My immediate answer is no, the KNN classification function is not invariant when different scale factors are used for different dimensions.

The reason is that this scaling changes the geometry of the feature space. The Euclidean distance between two re-scaled points $x'_i$ and $x'_j$ becomes a weighted distance:

$$d(x'_i, x'_j) = \sqrt{\sum_{d=1}^{D} \left( \frac{x_{id}}{s_d} - \frac{x_{jd}}{s_d} \right)^2} = \sqrt{\sum_{d=1}^{D} \frac{1}{s_d^2} (x_{id} - x_{jd})^2}$$

Since each dimension $d$ is weighted differently by $1/s_d^2$, the relative distances between points can change. A point that was the 3rd nearest neighbor in the original space might become the 10th nearest in the scaled space. Because the set of K-nearest neighbors can change, the final classification from the majority vote can also change.

However, as a side thought, if all dimensions were scaled by the *same* factor, so $s_d = s$ for all $d$, then the classification would be invariant. All distances are just scaled by a constant factor $1/s$. This preserves the distance rankings, meaning the set of K-nearest neighbors for any point will be identical, and the classification will not change.

## Question 2: KNN and Missing Data

My idea is to modify the standard KNN distance calculation to handle the missing values. We can use the binary mask vector, $m_i$, to make sure we only compare the features that are actually present.

When we calculate the distance between a new point $x$ and a training point $x_i$, we should only consider the dimensions where both vectors have a mask value of 1. We can change the Euclidean distance formula like this:

$$d'(x, x_i) = \sqrt{\sum_{d=1}^{D} m_d \cdot m_{i,d} \cdot (x_d - x_{i,d})^2}$$

In this formula, the term $m_d \cdot m_{i,d}$ will be 1 only if the feature is present in both vectors. Otherwise, it is 0, so that dimension is ignored in the distance calculation.

After calculating this new distance for all training points, the rest of the KNN algorithm is the same. We find the $K$ points with the smallest distance and use a majority vote to decide the class.

My observation is that this might not work well if there are too many missing values, because then we would be comparing points based on very few features. Another thought is to maybe fill the missing values with the mean of the feature, but that might add some bias to the data.

## Question 3: Probabilistic KNN

My approach is to prove that the function satisfies the two required conditions for a valid probability mass function (PMF): 1) all probabilities are non-negative, and 2) the probabilities sum to one over all possible outcomes.

**1. Non-negativity:** We need to show $P_{KNN}(Y = y|X = x) \geq 0$.

- The denominator $\epsilon C + K$ is positive, since the number of classes $C \geq 1$ and the number of neighbors $K \geq 1$.
- The indicator function $I(y_i = y)$ is either 0 or 1, so its sum $\sum_{i \in N_K(x)} I(y_i = y)$ is non-negative.
- The numerator $\epsilon +$ (a non-negative sum) must therefore be positive.

Since we are dividing a positive number by another positive number, the result is always positive. The non-negativity condition holds.

**2. Sum to one:** We need to show that $\sum_{y \in \mathcal{C}} P_{KNN}(Y = y|X = x) = 1$.

$$\sum_{y \in \mathcal{C}} P_{KNN}(Y = y|X = x) = \sum_{y \in \mathcal{C}} \frac{1}{\epsilon C + K} \left( \epsilon + \sum_{i \in N_K(x)} I(y_i = y) \right)$$

We pull out the constant term and split the summation.

$$= \frac{1}{\epsilon C + K} \left[ \sum_{y \in \mathcal{C}} \epsilon + \sum_{y \in \mathcal{C}} \left( \sum_{i \in N_K(x)} I(y_i = y) \right) \right]$$

Evaluating each part of the sum inside the brackets.

- The first term is simple: $\sum_{y \in \mathcal{C}} \epsilon = \epsilon C$.
- For the second term, we can swap the order of summation:

$$\sum_{i \in N_K(x)} \sum_{y \in \mathcal{C}} I(y_i = y)$$

  The inner sum $\sum_{y \in \mathcal{C}} I(y_i = y)$ must equal 1, because for any given neighbor $i$, its label $y_i$ belongs to exactly one class in $\mathcal{C}$. Hence, the second term simplifies to $\sum_{i \in N_K(x)} 1 = K$.

Substituting these results back into the equation:

$$= \frac{1}{\epsilon C + K} [\epsilon C + K] = 1$$

Both conditions are satisfied, so the function is a valid PMF.

## Question 4: LDA Probabilistic Prediction

The main idea is to start with Bayes' theorem and substitute the specific definitions for the LDA prior and likelihood.

First, the foundation for this is Bayes' theorem:

$$P(Y = y|\mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x}|Y = y)P(Y = y)}{P(\mathbf{X} = \mathbf{x})}$$

Next, I'll build the numerator, $P(\mathbf{X} = \mathbf{x}|Y = y)P(Y = y)$, by combining the prior $P(Y = y) = \pi_y$ and the Gaussian likelihood:

$$P(\mathbf{X} = \mathbf{x}|Y = y) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_y)^T\Sigma^{-1}(\mathbf{x} - \mu_y)\right)$$

Then, the denominator, $P(\mathbf{X} = \mathbf{x})$, is found by summing the joint probability over all classes $k = 1, \ldots, C$.

$$P(\mathbf{X} = \mathbf{x}) = \sum_{k=1}^{C} \pi_k \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T\Sigma^{-1}(\mathbf{x} - \mu_k)\right)$$

Putting it all together, the constant term $\frac{1}{|2\pi\Sigma|^{1/2}}$ appears in the numerator and every term of the denominator, so it cancels out. This leaves the final formula:

$$P(Y = y|\mathbf{X} = \mathbf{x}) = \frac{\pi_y \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_y)^T\Sigma^{-1}(\mathbf{x} - \mu_y)\right)}{\sum_{k=1}^{C} \pi_k \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T\Sigma^{-1}(\mathbf{x} - \mu_k)\right)}$$

## Question 5: Optimization

My plan is to use analytic methods to find the minimizers of $f(x) = x^4 - 2x^2 + 1$.

**1. Find Critical Points**
First, I'll compute the first derivative of the function $f(x)$ and set it to zero.

$$f'(x) = 4x^3 - 4x = 4x(x^2 - 1) = 4x(x - 1)(x + 1)$$

Setting $f'(x) = 0$ gives three critical points: $x = -1$, $x = 0$, and $x = 1$.

**2. Classify Critical Points**
Next, I'll use the second derivative test. The second derivative is $f''(x) = 12x^2 - 4$.

- At $x = -1$: $f''(-1) = 12(-1)^2 - 4 = 8 > 0$, so this is a local minimum.
- At $x = 0$: $f''(0) = 12(0)^2 - 4 = -4 < 0$, so this is a local maximum.
- At $x = 1$: $f''(1) = 12(1)^2 - 4 = 8 > 0$, so this is a local minimum.

**3. Identify Global Minimizers**
We have two local minima at $x = -1$ and $x = 1$. The value of the function at these points is:

$$f(1) = (1)^4 - 2(1)^2 + 1 = 0$$

$$f(-1) = (-1)^4 - 2(-1)^2 + 1 = 0$$

Both local minima have a value of 0. Since $\lim_{x\to\pm\infty} f(x) = \infty$, the function increases without bound away from the critical points. Therefore, the local minima we found must be the global minima. The global minimizers are $x = -1$ and $x = 1$.

## Question 6: Heart Disease Classification

### a. Data Set Properties

After running the preprocessing steps in the notebook, I found the following properties for the dataset:

- Training data cases: 242
- Test data cases: 61
- Feature dimensions: 18

### b. Logistic Regression Performance

After evaluating it on both the training and test sets, I found the following error rates:

- Training error rate: 0.1240
- Test error rate: 0.1639

### c. K-Nearest Neighbors Performance

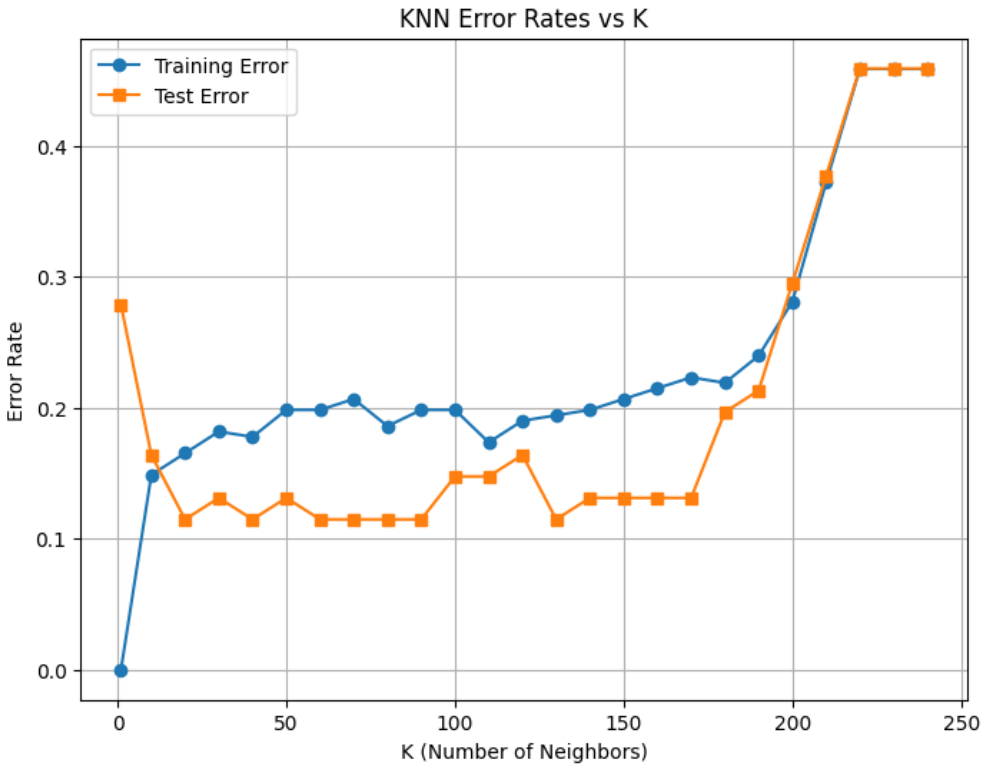


Figure 1: Training and test error rates for the KNN classifier as a function of the number of neighbors (K).

### d. Baseline LLM Performance

I ran the initial ‘llm.ipynb‘ notebook on a T4 GPU in Google Colab. This version used the simple starter prompt, which only provides the patient's age to the Falcon-H1-3B-Instruct model.

- Test error rate: 0.4587

### e. Prompt Engineering with More Features

My next idea was to see if providing more features would improve the LLM's performance. I experimented with two new prompt structures and compared them against the baseline on the training data:

- **Full Narrative:** A descriptive sentence that combines all features into a paragraph (e.g., "Age 52, sex: male, chest pain type: asymptomatic...").
- **Key-Value Concise:** A comma-separated list of feature names and their values (e.g., "age: 52, sex: male, cp: asymptomatic...").

The "Full Narrative" prompt performed best on the training data, achieving a training error of 0.3388. After selecting this as my best prompt, I ran it on the test set.

- Test error rate (Best Prompt): 0.2131

### f. Few-Shot Prompting

To give the small model more context, I tried a "few-shot" prompting strategy. My idea was to provide examples within the prompt itself to guide the model's classification. The prompt was structured to show the model three example patients without heart disease and three examples with heart disease, all taken from the training set, before presenting the actual test case for classification.

- Test error rate (Few-Shot Prompt): 0.3115

### g. Generative AI Usage

For Question 6 (parts d, e, and f), I utilized a generative AI assistant to help with the implementation in the ‘llm.ipynb‘ notebook. My use of the tool was focused on improving code efficiency and formatting. I prompted the assistant to help structure a batch processing loop to ensure the code would run without memory errors on the GPU. I also used it for minor syntax corrections and for help formatting the multi-line prompt strings.