Introduction to ML (CS771), 2024-2025-Sem-I	**Total Marks**	**25**
Quiz 1. August 20, 2024	**Duration**	**45 minutes**
Name	**Roll No.**	

Instructions:

1.	Clearly write your name (in block letters) and roll number in the provided boxes above.
2.	Write your final answers concisely in the provided space. You may use blue/black pen.
3.	We won't be able to provide clarifications during the quiz. If any aspect of some question appears ambiguous/unclear to you, please state your assumption(s) and answer accordingly.

Question 1: Write **T** or **F** for True/False in the box next to each question given below, with a brief (1-2 sentences at most) explanation in the provided space in the box below the question. Marks will be awarded only when the answer (T/F) and explanation <u>both</u> are correct. (**3 x 2 = 6 marks**)

1.1	Learning with Prototypes (LwP) based classification is, in general, faster at test time than *K*-nearest neighbors based classification	**T**

LwP only requires computing distance of the test input from the mean of each class, whereas *K*NN has to compute its distance from each of the training inputs.

1.2	Large Euclidean distance between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ implies their high similarity.	**F**

The expression of Euclidean distance is $\sqrt{\|a\|^2 + \|b\|^2 - 2a^\top b}$. If it is large then the dot product $a^\top b$ which measures their similarity should have a small value.

1.3	Prediction cost (time taken to predict the label of a test input) for a decision tree (DT) is proportional to the size of the training data that was used to construct the DT.	**F**

It is not true in general because we may have a very large amount of training data but the DT might still be simple (just requiring testing the values of a few features). We saw several examples in class where we only had to make one or two simple feature value tests.

Question 2: Answer the following questions concisely in the space provided below the question.

2.1	Consider two column vectors $\boldsymbol{a} = [a_1, a_2]$ and $\boldsymbol{b} = [b_1, b_2]$, each living in a two-dimensional vector space. Show that a similarity function defined as $k(\boldsymbol{a}, \boldsymbol{b}) = (1 + \boldsymbol{a}^\top \boldsymbol{b})^2$ is equivalent to the standard dot product but in a new vector space with both vectors transformed by a mapping function ϕ, i.e., $k(\boldsymbol{a}, \boldsymbol{b}) = \phi(\boldsymbol{a})^\top \phi(\boldsymbol{b})$. Clearly specify the form of the mapping ϕ. What is the dimensionality of this new vector space? **(4 marks)**

$$k(\boldsymbol{a}, \boldsymbol{b}) = (1 + \boldsymbol{a}^\top \boldsymbol{b})^2 = (1 + a_1 b_1 + a_2 b_2)^2$$
$$= 1 + a_1^2 b_1^2 + a_2^2 b_2^2 + 2a_1 b_1 + 2a_2 b_2 + 2a_1 a_2 b_1 b_2$$
$$k(\boldsymbol{a}, \boldsymbol{b}) = \left(1, a_1^2, a_2^2, \sqrt{2}a_1, \sqrt{2}a_2, \sqrt{2}a_1 a_2\right)^\top \left(1, b_1^2, b_2^2, \sqrt{2}b_1, \sqrt{2}b_2, \sqrt{2}b_1 b_2\right) = \phi(\boldsymbol{a})^\top \phi(\boldsymbol{b})$$

Thus $\phi(\boldsymbol{a}) = \left(1, a_1^2, a_2^2, \sqrt{2}a_1, \sqrt{2}a_2, \sqrt{2}a_1 a_2\right)$, and similar mapping for $\boldsymbol{b}$

The new vector space has dimensionality of 6

2.2 To speed up distance computations in K-nearest neighbors (KNN), consider two methods to reduce the dimensionality of the inputs: (1) Selecting 10 features (say using some feature selection method) from the originally given features, and (2) Applying some transformation such that each input is converted into a 10-dimensional binary vector. Which of these two methods would potentially be more suited for faster predictions in KNN, and why? **(3 marks)**

It will be faster with the binary vector since the distance between two binary vectors can be done very efficiently (which, in this case, requires just 10 bit-wise comparisons which can be done efficiently in hardware too) as compared to computing Euclidean distance between two 10-dimensional real-valued vectors.

2.3 Suppose we want to learn a decision tree (DT) from some toy training data shown on the right side. Each row is a training example has 3 continuous-valued features and a binary label. Show a suitable DT given this training data and briefly justify why your DT would be ideal for this data. **(3 marks)**

F1	F2	F3	Y
1	2	3	-1
2	4	6	-1
3	6	9	+1
4	8	12	+1

Note that the three features are correlated (just scaled versions of each other), so using just one feature is enough and the other two won't provide us any additional information. We can use the first feature (F1) and use a threshold equal to the mid-point between 2 and 3, i.e., 2.5 as the value to split on. It will result in a perfectly pure split (2 negatives on one leaf node and 2 positives on the other leaf node). The DT will simply have one node to test F1.

2.4 If prediction speed is not an issue, can internal nodes of a DT split data based on ALL features but not using criteria such as information gain? Brief explain your answer. **(3 marks)**

If we want to use all the features at an internal node to decide the splits, we can use any classification model (like LwP or nearest neighbor or any of your favorite classifier including a deep neural network!) which uses all the features to classify the data into the outgoing branches of this internal node.

2.5 What effect would minimizing a regularized loss function $L(\boldsymbol{w}) + \lambda R(\boldsymbol{w})$ have on the weight vector $\boldsymbol{w} \in \mathbb{R}^D$ if $R(\boldsymbol{w})$ is an ℓ_2 squared norm, i.e., $R(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{w}$, and why? **(3 marks)**

$R(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{w} = \sum_{i=1}^{D} w_i^2$. So minimizing the regularized loss function $L(\boldsymbol{w}) + \lambda R(\boldsymbol{w})$ not just make the training loss small but also the magnitude of the weight vector $\boldsymbol{w}$ small, which in turn will result in the individual entries of $\boldsymbol{w}$ to take small values.

2.6 Suppose you have trained two classifiers- LwP and DT - on the same training data with 5 classes. In future, suppose training data from 3 more classes becomes available (so now we have 8 classes). Which of these two models will be easier to retrain, and why? **(3 marks)**

For LwP, we will only need to compute the means of the 3 new classes. The means of the other classes won't need any updates. That's a neat aspect of LwP.

On the other hand, for a DT, it will require a lot more work because the split rules at root and the various internal nodes will have to be updates. Basically, this will mean learning the DT all over again using the training data of all 8 classes. ☹