Name:

Roll No.: **Dept.:**

IIT Kanpur
CS771A (IML)
Quiz-2
Date: Sept 12, 2023

Instructions: *Total:* **30 marks**

1. Duration is 45 minutes. Please write your name, roll number, department on **all pages**.
2. Write your answers clearly in the provided box. Keep your answer precise and concise.

Section 1 (Short/medium-length answer questions: 30 marks). .

1. Is a 3D sphere with a hole inside it a convex set? Briefly justify your answer. **(2 marks)**

   No, because it is possible to pick two points inside the sphere such that the line joining them will have a segment that falls inside the hole. The points on that segment of the line aren't in the same set as those two points.

2. Briefly explain (in at most 2-3 sentences) how AdaGrad defines a different learning rate in each dimensions (it is okay if you can't write the exact expression; explaining in words is fine). **(2 marks)**

   For each dimension, AdaGrad uses (inverse of) the sum of squares of previous gradients of that dimension to define the learning rate for that dimension.

3. Briefly explain (in at most 2-3 sentences) why mini-batch gradient descent is better than stochastic gradient descent which uses a single training example in each iteration. **(2 marks)**

   Mini-batch GD uses the average gradient computed using $B > 1$ data points in each mini-batch. This averaging done in min-batch GD helps reduce the variance of the approximate gradient as compared to stochastic GD which computes an approximate gradient using a single data point.

4. We have an optimization problem in which we have a non-negativity constraint on a $D \times 1$ variable of interest $\boldsymbol{w}$. From a computational cost point of view, which of the following two approaches will be better and why: (1) projected gradient descent, and (2) Lagrangian based optimization? **(2 marks)**

   Non-negativity constraint is easier and faster to handle with projected GD since we only need to perform standard GD followed by a thresholding (setting the negative values to zero). In contrast, Lagrangian approach will be more expensive since it requires introduces Lagrange multipliers for each constraints (we have D constraints, and solving the Lagrangian.

5. Gaussian (RBF) kernel is defined as $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\gamma||\boldsymbol{x}_n - \boldsymbol{x}_m||^2)$ where $\gamma > 0$ is the bandwidth parameter. Suppose we want each feature of the original inputs $\boldsymbol{x}_n, \boldsymbol{x}_m \in \mathbb{R}^D$ to have a different bandwidth parameter. Write down the expression for this variant of the RBF kernel. **(2 marks)**

   Note that the RBF kernel can be written as $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\sum_{d=1}^{D} \gamma(x_{nd} - x_{md})^2)$. To use a different bandwidth for each feature, we can define this variant: $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\sum_{d=1}^{D} \gamma_d(x_{nd} - x_{md})^2)$, where γ_d is the bandwidth associated with the d^{th} feature.

6. What advantage does a linear SVM classifier offer over a Perceptron linear classifier? Can you modify the Perceptron algorithm to have a similar effect? If yes, how? If no, why not? **(2 marks)**

   Linear SVM will learn the hyperplane with the largest possible margin. In contrast, Perceptron will learn any hyperplane that separates the classes but with no guarantee that the margin will be high. If we want Perceptron's hyperplane to also have a large margin, we can change the mistake condition from $\boldsymbol{w}^\top \boldsymbol{x}_n + b < 0$ to $\boldsymbol{w}^\top \boldsymbol{x}_n + b < \gamma$, which basically expresses our condition we want the training examples to not just be on the correct side of the hyperplane, but also be at least a certain distance away from it.

Name:

Roll No.: **Dept.:**

IIT Kanpur
CS771A (IML)
Quiz-2
Date: Sept 12, 2023

7. Consider two inputs $\boldsymbol{x} = [x_1, x_2]$ and $\boldsymbol{z} = [z_1, z_2]$. For the quadratic kernel defined as $k(\boldsymbol{x}, \boldsymbol{z}) = (1+\boldsymbol{x}^\top \boldsymbol{z})^2$, write down the expression of the feature mapping $\phi(\boldsymbol{x})$. **(2 marks)**

   Note that $k(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x}^\top \boldsymbol{z})^2 = (1 + x_1z_1 + x_2z_2)^2$. Expanding further, we get $k(\boldsymbol{x}, \boldsymbol{z}) = (1 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1 + 2x_2z_2 + 2x_1x_2z_1z_2)$. Comparing this expansion with $\phi(\boldsymbol{x})^\top\phi(\boldsymbol{z})$, we can see that $k(\boldsymbol{x}, \boldsymbol{z})$ is a dot product of two vectors $[1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2]$ and $[1, z_1^2, z_2^2, \sqrt{2}z_1, \sqrt{2}z_2, \sqrt{2}z_1z_2]$. Thus $\phi(\boldsymbol{x}) = [1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2]$.

8. Consider the learning with prototypes (LwP) model. Suppose we want to replace the Euclidean distance used in standard LwP by a kernelized distance using a kernel function k. For this kernelized version of LwP, write down the expression for the distance of a test point $\boldsymbol{x}_*$ from the mean of any class c where $c = 1, 2, \ldots, K$ and K is the total number of classes. The expression must only be in terms of the kernel function k and the training and test inputs. **(6 marks)**

   Assume the feature mapping for the kernel k to be ϕ. The mean of class c is given by $\phi(\boldsymbol{\mu}_c) = \frac{1}{N_c}\sum_{i:y_i=c}\phi(\boldsymbol{x}_i)$. Note that it is the mean of the ϕ representation of the inputs from class c.

   In the kernelized version, the squared Euclidean distance of this mean from the test input $\boldsymbol{x}_*$ will be

$$
\begin{aligned}
||\phi(\boldsymbol{\mu}_c) - \phi(\boldsymbol{x}_*)||^2 &= ||\phi(\boldsymbol{\mu}_c)||^2 + ||\phi(\boldsymbol{x}_*)||^2 - 2\phi(\boldsymbol{\mu}_c)^\top\phi(\boldsymbol{x}_*) \\
&= \phi(\boldsymbol{\mu}_c)^\top\phi(\boldsymbol{\mu}_c) + \phi(\boldsymbol{x}_*)^\top\phi(\boldsymbol{x}_*) - 2\phi(\boldsymbol{\mu}_c)^\top\phi(\boldsymbol{x}_*) \\
&= \frac{1}{N_c^2}\sum_{i:y_i=c}\sum_{j:y_j=c} k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \frac{2}{N_c}\sum_{i:y_i=c} k(\boldsymbol{x}_i, \boldsymbol{x}_*)
\end{aligned}
$$

9. For a binary classification problem on the same training data, will of these will have a larger margin: (1) Hard-margin linear SVM, (2) Soft-margin SVM? Briefly justify your answer. **(2 marks)**

   Soft-margin SVM will have a larger margin because it allows some of the training examples to fall inside the margin region (or even fall on the wrong side of the hyperplane). As a result, the margin region in soft-margin SVM can be wider as compared to hard-margin SVM in which no training examples are allowed to fall within the margin region (and, consequently, the margin will be narrower).

10. Why is kernel SVM slower than linear SVM at test time? **(2 marks)**

    Short answer: Linear SVM prediction requires computing just a dot product between the weight vector and the test input whereas kernel SVM requires computing the similarities between the test input and each of the training inputs. **Longer answer:** For linear SVM, the weight vector can be computed and stored explicitly as $\boldsymbol{w} = \sum_{i=1}^{N_s}\alpha_i y_i \boldsymbol{x}_i$ (N_s is the number of support vectors) which is a finite dimensional vector (of size D if inputs are D-dimensional) and the prediction for a new test input $\boldsymbol{x}_*$ can be done by simply computing $y_* = \text{sign}(\boldsymbol{w}^\top\boldsymbol{x}_* + b)$. On the other hand, for kernel SVM, the weight vector is of the form $\boldsymbol{w} = \sum_{i=1}^{N_s}\alpha_i y_i \phi(\boldsymbol{x}_i)$, which can't be stored since the ϕ mapping is usually very high dimensional. Thus the prediction requires computing $\text{sign}(\boldsymbol{w}^\top\phi(\boldsymbol{x}_*) + b) = \text{sign}(\sum_{i=1}^{N_s}\alpha_i y_i \phi(\boldsymbol{x}_i)^\top\phi(\boldsymbol{x}_*) + b) = \text{sign}(\sum_{i=1}^{N_s}\alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}_*) + b)$, which is much more expensive.

11. **(MCQ)** For which of the following problems, the objectives function is convex? (1) Hard-margin linear SVM, (2) Soft-margin linear SVM, (3) Hard-margin kernel SVM, (4) Soft-margin kernel SVM. **(2 marks)**
    All four problems have convex objectives. Objective of hard-margin linear/kernel SVM is the squared norm of $\boldsymbol{w}$ which is convex, whereas the objective of soft-margin linear/kernel SVM is the squared norm of $\boldsymbol{w}$ plus the hinge loss, which is again convex. Margin constraints on the training examples are all linear functions of $\boldsymbol{w}$, and thus are also convex. Note: For all these four cases, in the dual, we solve a maximization problem which is *concave* but remember that from a minimization perspective, all the problems are convex.

Name:

Roll No.: **Dept.:**

IIT Kanpur
CS771A (IML)
Quiz-2
Date: Sept 12, 2023

12. **(MCQ)** Which of the following is/are true regarding the Newton's method for minimizing a function: (1) Its per iteration cost is higher than gradient descent, (2) It always gives the global minima, (3) It uses a linear approximation of the function being minimized, (4) It usually takes fewer iterations to converge as compared to gradient descent. **(2 marks)**

13. **(MCQ)** Which of the following statements about optimization methods is/are true: (1) Co-ordinate descent does not require gradients, (2) Newton's method does not require learning rate, (3) Subgradient descent automatically handles constraints (e.g., nonnegativity) on the variables being optimized over, (4) How quickly (in terms of number of iterations) gradient descent converges for convex functions depends on the learning rate. **(2 marks)**

Name:

Roll No.: **Dept.**:

IIT Kanpur
CS771A (IML)
Quiz-2
Date: Sept 12, 2023