Homework 3: Gradient Descent and Least Squares

Due November 2, 2023 (100 points)

Problem 1 (25 points)**.** Suppose $X \in \mathbb{R}^{d\times n}$ is a data matrix with $d > n$, and let $\mathbf{y} \in \mathbb{R}^n$ be a vector of targets. We would like to find a weight vector $\mathbf{w} \in \mathbb{R}^d$ that solves $X^\top \mathbf{w} = \mathbf{y}$. Unlike our typical setup in linear regression, the system $X^\top \mathbf{w} = \mathbf{y}$ is now *underdetermined*: It may have infinitely many solutions.

In this problem, we will derive an expression for the *minimum 2-norm* solution to $X^\top \mathbf{w} = \mathbf{y}$. Assume throughout this problem that the columns of X are linearly independent.

(a) Set up an optimization problem for finding the minimum 2-norm solution $\mathbf{w}^*$ satisfying $X^\top \mathbf{w} = \mathbf{y}$.

(b) Using the method of Lagrange multipliers, find an expression for $\mathbf{w}^*$ in terms of X and $\mathbf{y}$. You do *not* need the SVD for this problem.

(c) Any solution to $X^\top \mathbf{w} = \mathbf{y}$ has the form $\mathbf{w} = \mathbf{w}^* + \mathbf{w}_0$, where $\mathbf{w}^*$ is the solution we found earlier and $\mathbf{w}_0 \in \ker(X^\top)$. Use this fact to give an alternative proof that $\mathbf{w}^*$ is the minimum-norm solution to $X\mathbf{w} = \mathbf{y}$.

(d) Suppose we factor $X = QR$, where $Q \in \mathbb{R}^{d\times n}$ and $R \in \mathbb{R}^{n\times n}$. Not counting the time needed to compute the factorization, what is the big-O complexity of computing $\mathbf{w}^*$?

Problem 2 (25 points)**.** We now show that gradient descent on $\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{2}\|X^\top \mathbf{w} - \mathbf{y}\|^2$, *without* Tikhonov regularization, converges to the minimum-norm solution of $X^\top \mathbf{w} = \mathbf{y}$.

(a) We will use a constant step size for simplicity. Derive a constant step size t that guarantees that gradient descent converges to a minimizer of f.

(b) Suppose we initialize gradient descent at $\mathbf{w}_0 = \mathbf{0} \in \mathbb{R}^d$. Show that every subsequent iterate $\mathbf{w}_k$ is contained in $\operatorname{im}(X)$.

(c) Argue that the iterates $\mathbf{w}_k$ converge to a minimum-norm solution to $X^\top \mathbf{w} = \mathbf{y}$.

Problem 3 (20 points)**.** Given a collection of n data points $\mathbf{x}_i \in \mathbb{R}^d$, one way to obtain an underdetermined regression problem is to map each column $\mathbf{x}_i$ to a new point $\phi(\mathbf{x}_i) \in \mathbb{R}^D$ for $D > n$ and instead solve $\phi(X)\mathbf{w} = \mathbf{y}$, where $\phi(X) \in \mathbb{R}^{D\times n}$ is the matrix whose i-th column is $\phi(\mathbf{x}_i)$.

In Problem 5(d) of HW1, we showed that one can solve this regression problem while only evaluating the function $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$. In particular, letting $K_{\mathbf{xx}} \in \mathbb{R}^{n\times n}$ be the *kernel matrix* whose (i, j)-th entry is $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, we:

1. Solve $K_{\mathbf{xx}}\mathbf{c} = \mathbf{y}$ for $\mathbf{c} \in \mathbb{R}^n$.

2. Obtain new predictions $\mathbf{z} \in \mathbb{R}^d$ by computing the vector $\mathbf{k}_{\mathbf{z}} \in \mathbb{R}^n$ whose elements are $k(\mathbf{z}, \mathbf{x}_i)$ for $i \in \{1, \ldots, n\}$ and then computing $f(\mathbf{z}) = \mathbf{c}^\top \mathbf{k}_{\mathbf{z}}$.

This procedure is called *kernel regression*, and it works even if $\phi(\cdot)$ maps points $\mathbf{x} \in \mathbb{R}^d$ to *functions*: We can think of functions as infinite-dimensional vectors. It might be difficult to solve $\phi(X)^\top \mathbf{w} = \mathbf{y}$ directly when the "columns" of $\phi(X)$ are infinite-dimensional, but the kernel matrix $K_{\mathbf{xx}}$ is still $n \times n$! So long as we can compute the inner products $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, we can carry out kernel regression.

In this problem, we will derive an expression for $k(\mathbf{x}_i, \mathbf{x}_j)$ given the feature map $\phi : \mathbf{x} \mapsto k_{\mathbf{x}}(\cdot)$, where $k_{\mathbf{x}}(\cdot)$ is a *function* given by $k_{\mathbf{x}}(\mathbf{p}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{p}\|_2^2}{2\sigma^2}\right)$. This $k(\mathbf{x}_i, \mathbf{x}_j)$ is the well-known *radial basis function (RBF)* kernel.

(a) Propose an inner product $k_{\mathbf{x}_i}(\cdot) \cdot k_{\mathbf{x}_j}(\cdot)$ between *functions* $k_{\mathbf{x}_i}(\cdot)$ and $k_{\mathbf{x}_j}(\cdot)$.

Hint: The standard inner product on $\mathbb{R}^d$ is the "dot product" $\mathbf{u} \cdot \mathbf{v} = \sum_{k=1}^d u_k v_k$. What is an infinite-dimensional analog to this sum?

(b) Using your proposed inner product, provide a formula for computing $k(\mathbf{x}_i, \mathbf{x}_j) = k_{\mathbf{x}_i}(\cdot) \cdot k_{\mathbf{x}_j}(\cdot)$.

Problem 4 (30 points)**.** In this problem, we'll implement kernel regression using the RBF kernel derived in Problem 3 and use it to classify the well-known MNIST handwritten digits into classes "0", "1,",..., "9". We will learn 10 weight vectors $c_m \in \mathbb{R}^n$, each parametrizing a linear function $f_m(\mathbf{z}) = \mathbf{c}_m^\top \mathbf{k}_z$. To predict the class of a new handwritten digit $\mathbf{z}$, we compute $f_m(\mathbf{z})$ for each $m \in \{1, ..., 10\}$ and predict the class as $m^* = \text{argmax}_m f_m(\mathbf{z})$.

(a) Implement a function that computes the RBF kernel matrix $K_{\mathbf{xy}} \in \mathbb{R}^{n_x \times n_y}$ given $X \in \mathbb{R}^{n_x \times d}$ and $Y \in \mathbb{R}^{n_y \times d}$. The (i, j)-th entry of $K_{\mathbf{xy}}$ should be $k(\mathbf{x}_i, \mathbf{y}_j)$ for the RBF kernel you derived in Problem 3(b).

(b) Kernel regression entails solving $K_{\mathbf{xx}}\mathbf{c} = \mathbf{y}$ for a weight vector $\mathbf{c} \in \mathbb{R}^n$. As the MNIST training set contains 60,000 images, the kernel matrix $K_{\mathbf{xx}}$ is $60,000 \times 60,000$. This linear system may be difficult to directly solve on your laptop.

An alternative approach for large datasets is to solve $\min_{\mathbf{c}} \frac{1}{n}\|K\mathbf{c} - \mathbf{y}\|_2^2$ using *stochastic gradient descent* (SGD). Given an objective of the form $f(\mathbf{c}) = \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{c})$, an SGD iteration:

i) Randomly draws $L \leq N$ indices $i_1, ..., i_L$

ii) Approximates $f(\mathbf{c}_k)$ with $\hat{f}(\mathbf{c}_k) = \frac{1}{L}\sum_{\ell=1}^L f_{i_\ell}(\mathbf{c}_k)$

iii) Takes a gradient step with respect to $\hat{f}(\mathbf{c}_k)$

Assuming we sample L indices $i_1, \ldots, i_L$, derive an expression for the $k+1$-th SGD iterate $\mathbf{c}_{k+1}$ for the objective $f(\mathbf{c}) = \frac{1}{2}\|K\mathbf{c} - \mathbf{y}\|_2^2$. You can assume a constant step size t. What is the big-O complexity of computing $\mathbf{c}_{k+1}$ in terms of L and n?

(c) Implement SGD for solving the kernel regression problem on MNIST. Use the hyperparameters (batch size, step size, number of iterations, and kernel bandwidth $\frac{1}{2\sigma^2}$) that have been provided in the starter code. What is the accuracy of the kernel classifier on the test set?