CS 331: Algorithms and Complexity
Homework 4

Due date: November 4, 2026, end of day (11:59 PM), uploaded to Canvas.

Late policy: 15% off if submitted late, and 15% off for every further 24 hours before submission.

Please list all collaborators on the first page of your solutions.

When runtimes are unspecified, slower runtimes than the intended solution receive partial credit.

1 Problem 1

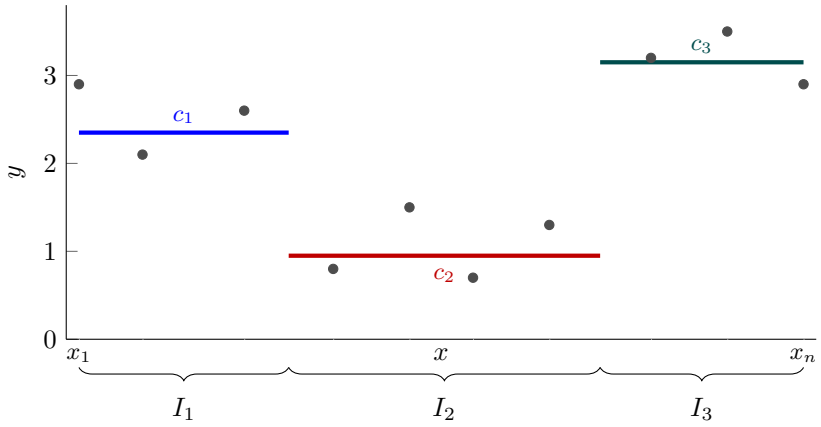
Let $f : [0, 1] \to \mathbb{R}$ be L-Lipschitz and strictly convex, and let $\epsilon > 0$.

(i) **(5 points)** Suppose that $\epsilon \geq \frac{L}{2}$. Give an algorithm that returns a point $\hat{x} \in [0, 1]$ such that $f(\hat{x}) - \min_{x \in [0,1]} f(x) \leq \epsilon$ using no queries to f, and $O(1)$ additional time.

(ii) **(15 points)** Suppose that $\epsilon < \frac{L}{2}$. Give an algorithm that returns a point $\hat{x} \in [0, 1]$ such that $f(\hat{x}) - \min_{x \in [0,1]} f(x) \leq \epsilon$ using $O(\log(\frac{L}{\epsilon}))$ queries to f, and $O(\log(\frac{L}{\epsilon}))$ additional time. Your algorithm should be based on *ternary search*.

2 Problem 2

Let $\{x_i \in \mathbb{R}\}_{i\in[n]}, \{y_i \in \mathbb{R}\}_{i\in[n]}$ be given with $x_1 < x_2 < \ldots < x_n$. For $k \in \mathbb{N}$, we call a function $f : [x_1, x_n] \to \mathbb{R}$ *k-piecewise-constant* (k-PC) if, for non-overlapping intervals $\{I_j\}_{j\in[k]}$ partitioning $[x_1, x_n]$, f is constant on each interval, i.e., for each $j \in [k]$, $f(x) = c_j$ for $c_j \in \mathbb{R}$ and all $x \in I_j$.



(i) **(5 points)** Give an algorithm that on inputs $\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}$, returns the c_1 defining a 1-PC $f : [x_1, x_n] \to \mathbb{R}$ that minimizes, over all 1-PC f,

$$\sum_{i\in[n]} (f(x_i) - y_i)^2 . \tag{1}$$

For full credit, your algorithm should run in time $O(n)$.

(ii) **(15 points)** Give an algorithm that on inputs $\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}$, and $C > 0$, returns the $\{c_j\}_{j\in[k]}$ and $\{I_j\}_{j\in[k]}$ defining a k-PC $f : [x_1, x_n] \to \mathbb{R}$ that minimizes, over all k-PC f, and all possible $k \in [n]$,

$$\sum_{i\in[n]} (f(x_i) - y_i)^2 + Ck. \tag{2}$$

The input $C > 0$ should be interpreted as the additional cost charged per piece used to define f (without this cost, we would always choose $k = n$ to minimize (2)).

For full credit, your algorithm should run in time $O(n^3)$.

3 Problem 3

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be a symmetric matrix. Prove the following bidirectional statements about $\mathbf{A}$.

(i) **(10 points)** $\mathbf{A}$ is positive semidefinite iff $\mathbf{A}_{S \times S}$ is positive semidefinite, for all $S \subseteq [d]$, where $\mathbf{A}_{S \times S}$ denotes the $|S| \times |S|$ submatrix formed by the entries $\{\mathbf{A}_{ij}\}_{(i,j) \in S \times S}$.

For example, below is an example of $\mathbf{A}$ and $\mathbf{A}_{S \times S}$ when $d = 3$ and $S = \{1, 2\}$:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 3 & -2 \\ 3 & -2 & 4 \end{pmatrix}, \quad \mathbf{A}_{S \times S} = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$$

(ii) **(10 points)** $\mathbf{A}$ is positive semidefinite iff it can be written in the form $\mathbf{A} = \mathbf{B}^\top \mathbf{B}$, for some (potentially asymmetric) matrix $\mathbf{B} \in \mathbb{R}^{d \times d}$.

No algorithm is necessary to solve this problem.

4 Problem 4

Let $\mathbf{v} \in \mathbb{R}^d$, and let $\mathbf{M} \in \mathbb{R}^{d \times d}$ be positive definite, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d > 0$ and corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d$. Recall that if $\mathbf{U} \in \mathbb{R}^{d \times d}$ has the $\{\mathbf{u}_i\}_{i \in [d]}$ as its columns, and $\mathbf{\Lambda} \in \mathbb{R}^{d \times d}$ is a diagonal matrix with the $\{\lambda_i\}_{i \in [d]}$ along the diagonal, we can write

$$\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \sum_{i \in [d]} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top.$$

In this problem, we analyze gradient descent for minimizing the high-dimensional quadratic

$$f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top \mathbf{M}\mathbf{x} - \mathbf{v}^\top \mathbf{x}.$$

In particular, we consider the gradient descent update

$$\mathbf{x}' \leftarrow \mathbf{x} - \frac{1}{\lambda_1}\nabla f(\mathbf{x}). \tag{3}$$

(i) **(5 points)** Compute the gradient $\nabla f(\mathbf{x})$. Why is $\mathbf{x}^\star := \mathbf{M}^{-1}\mathbf{v}$ the global minimizer to f?

(ii) **(10 points)** Suppose we start gradient descent at a point $\mathbf{x} \in \mathbb{R}^d$ with

$$\mathbf{x} - \mathbf{x}^\star = \mathbf{U}\mathbf{c} = \sum_{i \in [d]} c_i \mathbf{u}_i,$$

where $\mathbf{c} \in \mathbb{R}^d$ is a *coefficient vector* with entries $\{c_i\}_{i \in [d]}$. Also, after the update (3), let

$$\mathbf{x}' - \mathbf{x}^\star = \mathbf{U}\mathbf{c}' = \sum_{i \in [d]} c_i' \mathbf{u}_i,$$

where $\mathbf{c}' \in \mathbb{R}^d$ again has entries $\{c_i'\}_{i \in [d]}$. Prove that for all $i \in [d]$,

$$c_i' = \left(1 - \frac{\lambda_i}{\lambda_1}\right) c_i.$$

It may be helpful to rewrite $\nabla f(\mathbf{x}) = \mathbf{M}(\mathbf{x} - \mathbf{x}^\star)$ (explain why this is true if you do so).

(iii) **(5 points)** Conclude that if $\kappa := \frac{\lambda_1}{\lambda_d}$, we have $\|\mathbf{x}' - \mathbf{x}^\star\|_2 \leq (1 - \frac{1}{\kappa}) \|\mathbf{x} - \mathbf{x}^\star\|_2$.

5 Problem 5

(20 points) Complete the assignment at this link. This link is only accessible on your UT email.