Instructions

Submit your write-up on Gradescope as a neatly typeset (not scanned or handwritten) PDF document by 11:59 PM of the due date.

If you are working in a group, the group should produce a single write-up that is submitted by one of the group members. On Gradescope, the submitter must specify all of the group members upon submission of the write-up. More details can be found on the Gradescope Student Workflow help page (https://gradescope.com/help#help-center-section-student-workflow).

(Each group member should verify that the submission was successful.)

Make sure the names and UNIs of every member of your group appear prominently on the first page of your write-up.

On Gradescope, be sure to select the pages containing your answer for each problem. More details can be found on the Gradescope Student Workflow help page (https://gradescope.com/help#help-center-section-student-workflow). If you fail to select the pages containing your answer to a given problem, you will not receive any credit for that problem.

You are welcome to use the Markdown or \LaTeX source for the assignment as a template for your write-up. I use Pandoc (http://pandoc.org) to translate the Markdown to \LaTeX and ultimately to PDF.

Source code

Please combine all requested source code files into a single ZIP file, along with a plain text file called README that contains your name(s) and briefly describes all of the other files in the ZIP file. Do not include the data files. Submit this ZIP file on Courseworks. (Only one group member per group should do this.)

Clarity and precision

One of the goals in this class is for you to learn (i) to reason about machine learning problems and algorithms, and (ii) to make clear and precise claims and arguments about them.

A clear and precise argument is not the same as a long, excessively detailed argument. Unnecessary details and irrelevant side-remarks often make an argument less clear. And non-factual statements also detract from the clarity of an argument.

Points may be deducted for answers and arguments that lack sufficient clarity or precision. Moreover, a best-effort (but time-economical) attempt will be made to understand such answers/arguments, and the grade you will receive will be based on the correctness of this understanding.

Problem 1

In this problem, you will practice verifying the convexity of certain functions.

(a) Let $\mathcal{D}$ be a finite non-empty subset of $\mathbb{R}^d$, and consider the statistical model $\{P_\theta : \theta \in \mathbb{R}^d\}$ for iid random variables $X_1, \ldots, X_n$, where the probability mass function for $X_1$ is given by

$$p_\theta(x) \propto \exp(\theta^\top x), \quad x \in \mathcal{D},$$

and $p_\theta(x) = 0$ for $x \notin \mathcal{D}$. Prove that for any $x_1, \ldots, x_n \in \mathcal{D}$, the log-likelihood function

$$\text{LL}(\theta) = \ln \left( \prod_{i=1}^n p_\theta(x_i) \right), \quad \theta \in \mathbb{R}^d$$

is concave (i.e., $-\text{LL}$ is convex).

(b) A twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is strictly convex if its second-derivative matrix at any $x \in \mathbb{R}^d$ is positive definite. Is the function $f : \mathbb{R} \to \mathbb{R}$ given by

$$f(x) = \exp(-x), \quad x \in \mathbb{R}$$

strictly convex? Explain (with a short proof) why or why not.

(c) Let $\theta \in \mathbb{R}^d$ be a non-zero vector. Is the function $f : \mathbb{R}^d \to \mathbb{R}$ given by

$$f(x) = \|x - \theta\|_2^2 + 2\theta^\top x + 1, \quad x \in \mathbb{R}^d$$

strictly convex? Explain (with a short proof) why or why not.

(d) Let $\theta \in \mathbb{R}^d$ be a non-zero vector. Is the function $f : \mathbb{R}^d \to \mathbb{R}$ given by

$$f(x) = \exp(\theta^\top x), \quad x \in \mathbb{R}^d$$

strictly convex? Explain (with a short proof) why or why not.

Your solution:

(a)

(b)

(c)

(d)
Problem 2

In this problem, you will consider an optimization problem about convex functions.

Suppose you have training examples \((x_1, y_1), \ldots, (x_n, y_n)\) from \(\mathbb{R}^1 \times \mathbb{R}\), and you would like to compute the minimum empirical squared loss risk on these examples,

\[
\min_{h \in \mathcal{H}} \hat{R}(h), \quad \hat{R}(h) := \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2,
\]

where \(\mathcal{H}\) denotes the set of all convex functions \(h : \mathbb{R} \rightarrow \mathbb{R}\).

We will do this by using an alternative definition of convex functions (related to the one given for differentiable functions). A function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) is convex if, for every \(x \in \mathbb{R}^d\), there exists a vector \(\lambda \in \mathbb{R}^d\) (possibly specific to \(x\)) such that

\[
f(x') \geq f(x) + \lambda^\top (x' - x), \quad x' \in \mathbb{R}^d.
\]

(The vector \(\lambda\) is said to be a subgradient of \(f\) at \(x\); in parts (b) and (c) below, we just need the \(d = 1\) case of this definition, in which \(\lambda\) is a scalar.)

(a) Prove that if a function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) is convex according to the definition given above, then for any random vector \(X\) taking values in \(\mathbb{R}^d\), we have \(f(EX) \leq E f(X)\). (This latter property is equivalent to the definition for convexity given in lecture.)

(b) Consider the optimization problem where the optimization variables are \(z_1, \ldots, z_n, g_1, \ldots, g_n \in \mathbb{R}\), the objective function is

\[
\frac{1}{n} \sum_{i=1}^{n} (z_i - y_i)^2,
\]

and the \(n^2 - n\) constraints are

\[
z_i \geq z_j + g_j (x_i - x_j), \quad 1 \leq i \neq j \leq n.
\]

Prove that this optimization problem is a convex optimization problem.

(c) Explain why the optimal value of the optimization problem from part (b) is the optimal value of the optimization problem \(\min_{h \in \mathcal{H}} \hat{R}(h)\) defined above.

Your solution:

(a)

(b)

(c)
Problem 3

In this problem, you will experimentally study convergence behavior of gradient descent for logistic regression.

Let \((x_1, y_1), \ldots, (x_n, y_n)\) be training examples from \(\mathbb{R}^d \times \{0, 1\}\) for a binary classification problem. The following optimization problem specifies the logistic regression MLE parameters (with explicit affine expansion):

\[
\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ \ln (1 + \exp(\beta_0 + x_i^\top \beta)) - y_i (\beta_0 + x_i^\top \beta) \right\}.
\]

(a) Give concise and unambiguous pseudocode for a gradient descent algorithm that approximately solves this optimization problem. Be explicit about how the gradients are computed. Assume the initial solution, step sizes, and number of iterations are provided as inputs.

(b) Implement the gradient descent algorithm from part (a), except use \(\beta_0 = 0\) and \(\beta = 0\) as the initial solution, choose the step sizes using a backtracking line search (with initial step size \(\eta = 1\)), and use as many iterations as are required to achieve a prescribed objective value. You can use library functions that implement standard linear algebraic operations and simple functions such as \(\exp\) and \(\log\); of course, you should not use (or look at the source code for) existing implementations of gradient descent or other optimization algorithms. Run your gradient descent code on the data set \texttt{hw3data.mat} from Courseworks (which has training features vectors and labels stored as \texttt{data} and \texttt{labels}, respectively). How many iterations are needed to achieve an objective value that is at most 0.65064?

(c) The feature vectors in the data set from \texttt{hw3data.mat} are three-dimensional, so they are (relatively) easy to inspect. Investigate the data by plotting it and/or computing some statistics about the features. Do you notice anything peculiar about the features? Use what you discover to design an invertible \textit{linear transformation} of the feature vectors \(x_i \mapsto Ax_i\) such that running gradient descent on this transformed data \((Ax_1, y_1), \ldots, (Ax_n, y_n)\) reaches an objective value of 0.65064 in (many) fewer iterations. Describe the steps and reasoning in this investigation, as well as your chosen linear transformation (as a \(3 \times 3\) matrix). How many iterations were required to achieve this stated objective value?

(d) Create a new version of your gradient descent code with the following changes.

1. Use only the first \([0.8n]\) examples to define the objective function; keep the remaining \(n - [0.8n]\) examples as a validation set.

2. Use the following stopping condition. After every power-of-two \((2^0, 2^1, 2^2, \text{etc.})\) iterations of gradient descent, record the \textit{validation error rate} (i.e., zero-one loss validation risk) for the linear classifier based on the current \((\beta_0, \beta)\). If this validation error rate is more than 0.99 times that of the best validation error rate previously computed, and the number of iterations executed is at least 32 (which is somewhat of an arbitrary number), then stop.

Run this modified gradient descent code on the original \texttt{hw3data.mat} data, as well as the linearly transformed data (from part (c)). In each case, report: (1) the number of iterations executed, (2) the final objective value, and (3) the final validation error rate.

Please submit your source code on Courseworks.

Your solution:

(a)

(b)

(c)

(d)

\footnote{The actual minimum value is less than 0.65064.}

\footnote{Normally you would not simply select the first \([0.8n]\) examples, but rather pick a random subset of \([0.8n]\) examples. But I have already randomized the order of the examples in \texttt{hw3data.mat}.}
Problem 4

In this problem, you will derive and implement a coordinate ascent algorithm for soft-margin kernel SVM.

The dual of the soft-margin kernel SVM problem is

$$\max_{\alpha_1, \ldots, \alpha_n \in \mathbb{R}} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j$$

subject to $0 \leq \alpha_i \leq C$, $i = 1, \ldots, n$.

Here, $(x_1, y_1), \ldots, (x_n, y_n)$ are the training examples from $\mathbb{R}^d \times \{-1, +1\}$, $K: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the kernel function, and $C > 0$ is the “trade-off parameter” in the soft-margin SVM problem.

A simple algorithm for solving this dual problem is dual coordinate ascent:

- Start with $\alpha_i := 0$ for all $i = 1, \ldots, n$.
- For $t = 1, 2, \ldots$ until some stopping condition is met:
  - Choose a permutation $\pi_t$ on $\{1, \ldots, n\}$.
  - For $i = \pi_t(1), \ldots, \pi_t(n)$: set $\alpha_i \in [0, C]$ to maximize the objective (with all other variables $\alpha_j$, $j \neq i$, held fixed), i.e.,
    $$\alpha_i \in \arg \max_{\alpha' \in [0, C]} g(\alpha_1, \ldots, \alpha_{i-1}, \alpha'_i, \alpha_{i+1}, \ldots, \alpha_n).$$

Often, the permutation $\pi_t$ is chosen uniformly at random.

The main challenge in implementing this algorithm is determining how to set $\alpha_i$ in the innermost loop. Fortunately, the objective—when viewed as only a function of $\alpha_i$—is a concave quadratic function of $\alpha_i$, and the constraints on $\alpha_i$ are very simple.

(a) Let $a, b, c, l, u \in \mathbb{R}$ be such that $a \geq 0$ and $l \leq u$. Consider the function $g: \mathbb{R} \to \mathbb{R}$ given by $g(z) = -az^2/2 + bz + c$. What is the maximizer of $g$ over the range $[l, u]$? Give your answer as a simple expression in terms of $a, b, c, l, u$. (You may want to use a conditional expression.)

(b) Use the result from (a) to derive a formula for $\alpha_i$ in the dual coordinate ascent algorithm, expressed in terms of the training examples, the trade-off parameter, and the other variables $\alpha_j$, $j \neq i$.

(c) Write a program that implements this dual coordinate ascent algorithm for the special case of the linear kernel $K(x, x') := x^\top x'$, trade-off parameter $C := 10/n$, and where the permutation $\pi_t$ in each iteration $t$ is the identity permutation, i.e., $\pi_t(i) = i$ for all $i = 1, \ldots, n$. You can use library functions that implement standard linear algebraic operations; of course, you should not use (or look at the source code for) existing implementations of coordinate ascent or other SVM solvers. Run your program on the data from hw3data.mat after replacing every 0 in labels with −1 and as well as the “standardization” feature transformation (from HW2). Report (1) the objective value after $t = 2$ iterations and (2) the weight vector $w := \sum_{i=1}^{n} \alpha_i y_i x_i$ after two iterations.

(d) What is the running time complexity of each iteration of the dual coordinate ascent algorithm, given as a function of $n$, the number of training examples? You can use Big O notation, and you may assume $d = 3$. Briefly justify your answer.

Please submit your source code on Courseworks.

Your solution:

(a)

(b)

(c)

(d)

\footnote{There is a typo here—there should really be a 1/2 somewhere in the objective function. But for this homework assignment, please just use the objective function (without any factor of 1/2) as given.}