# COMS 4771-2 F18 Homework 4 (due November 14, 2018)

## Instructions

Submit your write-up on Gradescope as a neatly typeset (not scanned nor handwritten) PDF document by 11:59 PM of the due date.

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 don't select the pages containing your answer to a problem, you'll receive a zero for that problem.)

Make sure **your name and your UNI** appears prominently on the first page of your write-up.

### Source code

Please combine all requested source code files into a *single* ZIP file[1], along with a plain text file called `README` that contains your name and briefly describes all of the other files in the ZIP file. **Do not include the data files.** Submit this ZIP file on Courseworks.

### Clarity and precision

One of the goals in this class is for you to learn to reason about machine learning problems and algorithms. To reason about these things, you must be able 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 time-economical attempt will be made to interpret such answers/arguments, and the grade you receive will be based on this interpretation.

[1] See https://en.wikipedia.org/wiki/Zip_(file_format).

# Problem 1 (30 points)

In this problem, you will study a case where maximum likelihood fails but empirical risk minimization succeeds.

Consider the following probability distribution $P$ on $\mathcal{X} \times \mathcal{Y}$, for $\mathcal{X} = \{0,1\}^2$ and $\mathcal{Y} = \{0,1\}$.

$P(X = (x_1, x_2))$ :

| | $x_1 = 0$ | $x_1 = 1$ |
|---|---|---|
| $x_2 = 0$ | $\frac{1-\epsilon}{2}$ | $\frac{1-\epsilon}{2}$ |
| $x_2 = 1$ | $\frac{\epsilon}{2}$ | $\frac{\epsilon}{2}$ |

$P(Y = 1 \mid X = (x_1, x_2))$ :

| | $x_1 = 0$ | $x_1 = 1$ |
|---|---|---|
| $x_2 = 0$ | 0 | 1 |
| $x_2 = 1$ | 1 | 0 |

Above, regard $\epsilon$ as a small positive number between 0 and 1. Let $f^\star$ be the Bayes optimal classifier for $P$. (In this problem, we are concerned with zero-one loss.)

Now also consider the statistical model $\mathcal{P}$ for $Y \mid X$: $\mathcal{P} = \{P_A, P_B\}$, where

$$P_A(Y = 1 \mid X = (x_1, x_2)) = \begin{cases} \frac{4}{5} & \text{if } x_1 = 0, \\ \frac{1}{5} & \text{if } x_1 = 1; \end{cases} \qquad P_B(Y = 1 \mid X = (x_1, x_2)) = \begin{cases} 0 & \text{if } x_1 = 0, \\ 1 & \text{if } x_1 = 1. \end{cases}$$

Note that $P \notin \mathcal{P}$, and that distributions in $\mathcal{P}$ do not specify the distribution of $X$ (like logistic regression). Let $f_A$ and $f_B$ be the Bayes optimal classifiers, respectively, for $P_A$ and $P_B$.

The maximum likelihood approach to learning a classifier selects the distribution in $\mathcal{P}$ of highest likelihood given training data (which are regarded as an iid sample), then returns the optimal classifier for the chosen distribution (i.e., $f_A$ if $P_A$ is the maximum likelihood distribution, otherwise $f_B$).

(a) Give simple expressions for the Bayes optimal classifiers for $P$, $P_A$, and $P_B$. E.g.,

$$f^\star(x) = \begin{cases} 1 & \text{if } x_1 + x_2 = 2, \\ 0 & \text{otherwise.} \end{cases}$$

(b) What is the risk of each classifier from Part (a) *under distribution $P$*?

(c) Suppose in the training data, the number of training examples of the form $((x_1, x_2), y)$ is equal to $N_{x_1,x_2,y}$, for $(x_1, x_2, y) \in \{0,1\}^3$. Give a simple rule for determining the maximum likelihood distribution in $\mathcal{P}$ in terms of $N_{x_1,x_2,y}$ for $(x_1, x_2, y) \in \{0,1\}^3$. Briefly justify your answer.

(d) If the training data is a typical iid sample from $P$ (with large sample size $n$), which classifier from Part (a) is returned by the maximum likelihood approach? Briefly justify your answer.

(e) If the training data is an iid sample from $P$ of size $n$, then how large should $n$ be so that, with probability at least 0.99, Empirical Risk Minimization returns the classifer in $\{f_A, f_B\}$ with smallest risk? Briefly justify your answer.

## Problem 2 (20 points)

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

$$p_\theta(x) = \frac{\exp(\theta^\top x)}{\sum_{y \in \mathcal{D}} \exp(\theta^\top y)}, \quad x \in \mathcal{D},$$

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

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

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

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

$$f(x) = \exp(-x) \quad \text{for all } x \in \mathbb{R}$$

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 $f\colon \mathbb{R}^d \to \mathbb{R}$ given by

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

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 $f\colon \mathbb{R}^d \to \mathbb{R}$ given by

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

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

## Problem 3 (20 points)

In this problem, you will formulate optimization problems as convex optimization problems using only "simple" objective and constraint functions.

Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ be given data. Write each of the following optimization problems as a convex optimization problem *in standard form* in which the objective function and constraint functions are linear or affine functions. You may introduce additional variables and constraints, but the number of variables and constraints should be no more than a linear function of $n$ and $d$. Briefly explain why the optimization problem you formulate has the same optimal solutions as the given one.

(a)

$$\min_{w \in \mathbb{R}^d} \quad \max_{i=1,\ldots,n} |x_i^\top w - y_i|.$$

(b)

$$\min_{w \in \mathbb{R}^d} \quad \sum_{i=1}^{n} \phi(x_i^\top w - y_i),$$

where $\phi\colon \mathbb{R} \to \mathbb{R}$ is the function defined by

$$\phi(t) := \begin{cases} 0 & \text{if } |t| \leq 1, \\ |t| - 1 & \text{if } |t| > 1. \end{cases}$$

(c)

$$\begin{aligned} \min_{w \in \mathbb{R}^d} \quad & \sum_{j=1}^{d} |w_j| \\ \text{s.t.} \quad & |x_i^\top w - y_i| \leq 1 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

## Problem 4 (30 points)

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):

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

(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 $\eta = 1$)[2], 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 `logreg.mat` from the course website (which has training features vectors and labels stored as `data` and `labels`, respectively). How many iterations of gradient descent are needed to achieve an objective value that is at most 0.65064?[3]

(c) The feature vectors in the data set from `logreg.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 *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 3×3 matrix). How many iterations of gradient descent 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 $\lfloor 0.8n \rfloor$ examples to define the objective function; keep the remaining $n - \lfloor 0.8n \rfloor$ examples as a validation set.[4]
2. Use the following stopping condition. After every power-of-two ($2^0$, $2^1$, $2^2$, etc.) iterations of gradient descent, record the *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 `logreg.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.

[2]Note that you should start with $\eta = 1$ in the backtracking line search for *each* step of gradient descent.

[3]The actual minimum value is less than 0.65064.

[4]Normally you would not simply select the first $\lfloor 0.8n \rfloor$ examples, but rather pick a random subset of $\lfloor 0.8n \rfloor$ examples. But I have already randomized the order of the examples in `logreg.mat`.