Submission instructions:

• If a problem asks for a numerical answer, you need only provide this answer. There is no need to show your work, unless you would like to.

• Please type up your solutions. We suggest using an online latex editor like www.overleaf.com, though this is not a requirement.

• Upload the PDF file for your homework to gradescope by midnight on Friday, Feb 22.

1. **Lines through the origin**. Suppose that we have data points \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\), where \(x^{(i)}, y^{(i)} \in \mathbb{R}\), and that we want to fit them with a line that passes through the origin. The general form of such a line is \(y = ax\): that is, the sole parameter is \(a \in \mathbb{R}\).

   (a) The goal is to find the value of \(a\) that minimizes the squared error on the data. Write down the corresponding loss function.

   (b) Using calculus, find the optimal setting of \(a\).

2. Suppose that \(y = x_1 + x_2 + \cdots + x_5 + z\), where:

   • \(x_1, \ldots, x_5, z \in \mathbb{R}\).

   • \(z\) is independent of all the \(x_i\)’s and has a Gaussian distribution with mean 1 and variance 1.

   (a) We wish to express \(y\) as a linear function of \(x_1, \ldots, x_5\). What is the linear function that minimizes MSE?

   (b) What is the mean squared error of the function in (a)?

3. We have a data set \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\), where \(x^{(i)} \in \mathbb{R}^d\) and \(y^{(i)} \in \mathbb{R}\). We want to express \(y\) as a linear function of \(x\), but the error penalty we have in mind is not the usual squared loss: if we predict \(\hat{y}\) and the true value is \(y\), then the penalty should be the absolute difference, \(|y - \hat{y}|\). Write down the loss function that corresponds to the total penalty on the training set.

4. We have \(n\) data points in \(\mathbb{R}^d\) and we want to compute all pairwise dot products between them. Show that this can be achieved by a single matrix multiplication.

5. Consider the following loss function on vectors \(w \in \mathbb{R}^4\):

   \[ L(w) = w_1^2 + 2w_2^2 + w_3^2 - 2w_3w_4 + w_4^2 + 2w_1 - 4w_2 + 4. \]

   (a) What is \(\nabla L(w)\)?

   (b) Suppose we use gradient descent to minimize this function, and that the current estimate is \(w = (0, 0, 0, 0)\). If the step size is \(\eta\), what is the next estimate?

   (c) What is the minimum value of \(L(w)\)?
(d) Is there a unique solution \( w \) at which this minimum is realized?

6. For each of the following functions of one variable, say whether it is convex, concave, both, or neither.

   (a) \( f(x) = x^2 \)
   (b) \( f(x) = -x^2 \)
   (c) \( f(x) = x^2 - 2x + 1 \)
   (d) \( f(x) = x \)
   (e) \( f(x) = x^3 \)
   (f) \( f(x) = x^4 \)
   (g) \( f(x) = \ln x \)

7. Show that the matrix \( M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \) is not positive semidefinite.

8. Show that the matrix \( M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \) is positive semidefinite.

9. We are given a set of data points \( x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d \), and we want to find a single point \( z \in \mathbb{R}^d \) that minimizes the loss function

\[
L(z) = \sum_{i=1}^{n} \| x^{(i)} - z \|^2.
\]

Use calculus to determine \( z \), in terms of the \( x^{(i)} \).

10. Consider the loss function for ridge regression (ignoring the intercept term):

\[
L(w) = \sum_{i=1}^{n} (y^{(i)} - w \cdot x^{(i)})^2 + \lambda \| w \|^2
\]

where \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \mathbb{R}\) are the data points and \( w \in \mathbb{R}^d \). There is a closed-form equation for the optimal \( w \) (as we saw in class), but suppose that we decide instead to minimize the function using local search.

   (a) What is \( \nabla L(w) \)?
   (b) Write down the update step for gradient descent.
   (c) Write down a stochastic gradient descent algorithm.

11. In this problem, you are going to build a logistic regression digit classifier using the same MNIST dataset you used in past homeworks. If you don’t already have them, first go and download the train, validation, and test sets from the class website. (For a description of the data format, please see the last problem of Homework 0.)

Since the label set for this task consists of ten possible digits, you will be building a multi-class logistic regression classifier. The conditional probability of the digit label, \( y \), given the image feature vector, \( x \), when using this model class can be written as:

\[
P_w(y|x) = \frac{\exp(w_y \cdot x)}{\sum_{y'} \exp(w_{y'} \cdot x)}
\]
If the image feature vector is $d$-dimensional (i.e. $x \in \mathbb{R}^d$), then, for each digit type $y \in \{0, 1, 2, \ldots, 9\}$, the model has a $d$-dimension parameter vector, $w_y \in \mathbb{R}^d$. The role of each $w_y$ in this multi-class model is to determine the linear score, $(w_y \cdot x)$, of the corresponding digit, $y$, paired with feature vector, $x$. Ideally, once trained, the score of the correct digit will be higher than the scores of the rest of the digits, given the input, $x$. We can write the full set of parameters as $w = (w_0, w_1, \ldots, w_9)$, which we can view as a $(10 \cdot d)$-dimensional vector (i.e. all digit-specific vectors appended together).

(a) Before actually coding up this model, we need write down the objective function. Let our training data be denoted as a sequence of pairs: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})$. Here, $x^{(i)}$ is the image feature vector of the $i$th training datum, and $y^{(i)}$ is the correct digit label. Write maximum likelihood estimation for the multi-class logistic regression model on this training set in the form of a minimization problem, including a term for L2 regularization. Use $\lambda$ to denote the hyperparameter that is the weight of the regularization term. Use log-likelihood rather than likelihood. This will help numerical stability when you implement your system. [Hint: In other words, if training consists of finding $w$ to minimize the function $L(w)$, what should $L(w)$ be?]

(b) Write the equation for $\nabla L(w)$. Remember to include the regularization term. [Hint: you can always just differentiate $L(w)$ w.r.t. each dimension of $w$ separately and then collect the results in a vector. However, it may be faster if you try to use vector calculus directly, computing the gradient w.r.t. each class-specific vector $w_y$ in turn.]

(c) Now it’s time to begin coding. We recommend using PyTorch, but you can use any language you want. You are going to train your first logistic regression classifier.

First, you have to define a feature representation of the input images, i.e. you have to define the dimensions of $x$. For this part, we will use the following simple feature representation: Let $x$ be 57-dimensional. The first 28 dimensions of $x$ will contain the number of dark pixels in each row of the input image (i.e. $x_i$ will be the number of pixels that have a value above or below some threshold, $\epsilon$, in row $i$ of the image). Similarly, let the next 28 dimensions of $x$ contain the number of dark pixels in each column. Finally, the last dimension of $x$ will be the intercept term, which will always have value 1. Implement a data loader that maps each image to a vector $x$ using this feature representation.

Next, you need to implement gradient descent to learn the model parameters, $w$, from the training set. If you are using PyTorch, it is possible to simply implement $L(w)$ and let PyTorch compute the gradients automatically. Alternatively, you can implement your equation (from part (b)) for the gradient directly. Either way, you will need to construct a training loop that iteratively updates $w$ based on the current gradient and a step-size hyperparameter, $\eta$. You may need to experiment with several settings of $\eta$ in order to find a step size that works on this dataset and objective.

Feel free to get advice on your implementation from other students and from TAs, but make sure that you write your own code. You may need to do some debugging to get gradient descent to work properly. We recommend printing out the value of the training objective after each iteration, as well as the accuracy of the classifier on the validation set. This will help you find appropriate hyperparameters, e.g. $\eta$, $\lambda$, and $\epsilon$. You may encounter issues with numerical stability depending on how you implement your gradient computation. Remember, directly exponentiating large values (or large negative values) will lead to unstable results. Use alternative formulas involving logs when possible.

2-3
Finally, for this part, we want you to plot (or simply list the values of) the objective function as it changes over 20 iterations of gradient descent for the best hyperparameter settings you found. Which values of $\lambda$, $\eta$, and $\epsilon$ lead to the best validation accuracy? What validation accuracy were you able to achieve? [Note: using our simple feature representation, your classification accuracy won’t be great but should be better than chance.]

(d) Now, implement stochastic gradient descent. Your step size, $\eta$, will likely need to be smaller than it was for standard gradient descent. Plot (or list the values of) the objective after each of 10 epochs of stochastic gradient descent for the best hyperparameter settings you can find. Does stochastic gradient descent lead to faster training?

(e) Finally, we’re going to turn you loose to define your own feature representations. Feel free to define any new features you like. It’s also fine to discard the old features, and design better ones from scratch. Find new features that improve your best validation accuracy from part (d) by at least 0.5%. This should not be difficult. Much better gains are possible. [Hint: You may need to try new hyperparameters or use a larger number of gradient descent steps depending on the features you built. Feel free to use more advanced optimization methods, e.g. Adagrad or LBFGS. Also, in general, features that consider ‘edges’ in the image are typically quite helpful.]

Describe the new features you implemented and report the best validation accuracy you achieved. Also include a plot (or list of values) of your training objective over the first 20 iterations. If you used a more advanced optimization method, tell us about it.