Homework #2

CSE 446/546: Machine Learning
Profs. Simon Du and Sewoong Oh
Due: **Wednesday** May 5, 2021 11:59pm

Please review all homework guidance posted on the website before submitting to Gradescope. Reminders:

- Make sure to read the "What to Submit" section following each question and include all items.
- Please provide succinct answers and supporting reasoning for each question. Similarly, when discussing experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. All explanations, tables, and figures for any particular part of a question must be grouped together.
- For every problem involving generating plots, please include the plots as part of your PDF submission.
- When submitting to Gradescope, please link each question from the homework in Gradescope to the location of its answer in your homework PDF. Failure to do so may result in deductions of up to *[5 points]* . For instructions, see `https://www.gradescope.com/get_started#student-submission`.
- Please recall that B problems, indicated in boxed text, are only graded for 546 students, and that they will be weighted at most 0.2 of your final GPA (see website for details). In Gradescope there is a place to submit solutions to A and B problems separately. You are welcome to create just a single PDF that contains answers to both, submit the same PDF twice, but associate the answers with the individual questions in Gradescope.
- If you collaborate on this homework with others, you must indicate who you worked with on your homework. Failure to do so may result in accusations of plagiarism.
- For every problem involving code, please include the code as part of your PDF for the PDF submission *in addition to* submitting your code to the separate assignment on Gradescope created for code. Not submitting all code files will lead to a deduction of *[1 points]* .

Not adhering to these reminders may result in point deductions.

Conceptual Questions *[10 points]*

A0. The answers to these questions should be answerable without referring to external materials. Briefly justify your answers with a few words.

a. *[2 points]* Suppose that your estimated model for predicting house prices has a large positive weight on the feature `number of bathrooms`. If we remove this feature and refit the model, will the new model have a strictly higher error than before? Why?

b. *[2 points]* Compared to L2 norm penalty, explain why a L1 norm penalty is more likely to result in sparsity (a larger number of 0s) in the weight vector.

c. *[2 points]* In at most one sentence each, state one possible upside and one possible downside of using the following regularizer: $\left(\sum_i |w_i|^{0.5}\right)$

d. *[1 points]* True or False: If the step-size for gradient descent is too large, it may not converge.

e. *[2 points]* In your own words, describe why stochastic gradient descent (SGD) works, even though only a small portion of the data is considered at each update.

f. *[2 points]* In at most one sentence each, state one possible advantage of SGD over GD (gradient descent), and one possible disadvantage of SGD relative to GD.

What to Submit:

- For each part a-f, 1-2 sentences containing your answer.

Convexity and Norms

A1. A *norm* $\|\cdot\|$ over $\mathbb{R}^n$ is defined by the properties: (*i*) non-negativity: $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$ with equality if and only if $x = 0$, (*ii*) absolute scalability: $\|a\,x\| = |a|\,\|x\|$ for all $a \in \mathbb{R}$ and $x \in \mathbb{R}^n$, (*iii*) triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

a. *[3 points]* Show that $f(x) = (\sum_{i=1}^n |x_i|)$ is a norm. (Hint: for (*iii*), begin by showing that $|a+b| \leq |a| + |b|$ for all $a, b \in \mathbb{R}$.)

b. *[2 points]* Show that $g(x) = \left(\sum_{i=1}^n |x_i|^{1/2}\right)^2$ is not a norm. (Hint: it suffices to find two points in $n = 2$ dimensions such that the triangle inequality does not hold.)

Context: norms are often used in regularization to encourage specific behaviors of solutions. If we define $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$ then one can show that $\|x\|_p$ is a norm for all $p \geq 1$. The important cases of $p = 2$ and $p = 1$ correspond to the penalty for ridge regression and the lasso, respectively.

What to Submit:

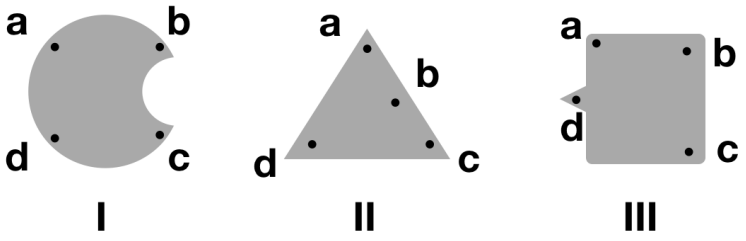
- For each part (a,b), a proof of the statement.

B1. *[6 points]* For any $x \in \mathbb{R}^n$, define the following norms: $\|x\|_1 = \sum_{i=1}^n |x_i|$, $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$, $\|x\|_\infty := \lim_{p\to\infty} \|x\|_p = \max_{i=1,\ldots,n} |x_i|$. Show that $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$.

What to Submit:

- A proof.

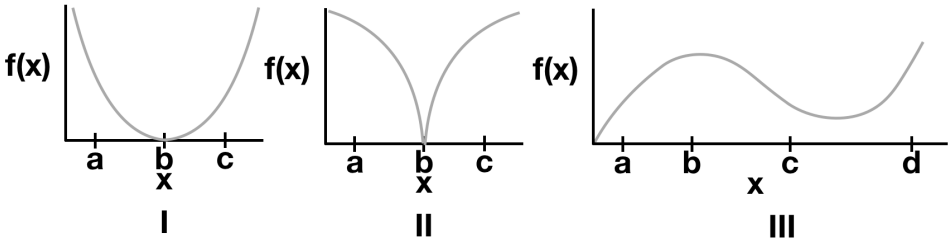
A2. *[3 points]* A set $A \subseteq \mathbb{R}^n$ is *convex* if $\lambda x + (1-\lambda)y \in A$ for all $x, y \in A$ and $\lambda \in [0, 1]$.
For each of the grey-shaded sets below (I-III), state whether each one is convex, or state why it is not convex using any of the points a, b, c, d in your answer.



What to Submit:

- For each part (I, II, III), 1-2 sentences.

A3. *[4 points]* We say a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex on a set A if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ for all $x, y \in A$ and $\lambda \in [0, 1]$. For each of the functions shown below (I-III), state whether each is convex on the specified interval, or state why not with a counterexample using any of the points a, b, c, d in your answer.



a. Function in panel I on $[a, c]$

b. Function in panel II on $[a, c]$

c. Function in panel III on $[a, d]$

d. Function in panel III on $[c, d]$

What to Submit:

- For each part a-d, 1-2 sentences.

B2. Use just the definitions above and let $\|\cdot\|$ be a norm.

a. *[3 points]* Show that $f(x) = \|x\|$ is a convex function.

b. *[3 points]* Show that $\{x \in \mathbb{R}^n : \|x\| \leq 1\}$ is a convex set.

c. *[2 points]* Draw a picture of the set $\{(x_1, x_2) \ : \ g(x_1, x_2) \leq 4\}$ where $g(x_1, x_2) = \left(|x_1|^{1/2} + |x_2|^{1/2}\right)^2$. (This is the function considered in 1b above specialized to $n = 2$.) We know g is not a norm. Is the defined set convex? Why not?

Context: It is a fact that a function f defined over a set $A \subseteq \mathbb{R}^n$ is convex if and only if the set $\{(x, z) \in \mathbb{R}^{n+1} : z \geq f(x), x \in A\}$ is convex. Draw a picture of this for yourself to be sure you understand it.

What to Submit:

- For each part (a,b), a proof of the statement.
- For (c), a picture of the set, and 1-2 sentences answering the question.

B3. For $i = 1, \ldots, n$ let $\ell_i(w)$ be convex functions over $w \in \mathbb{R}^d$ (e.g., $\ell_i(w) = (y_i - w^\top x_i)^2$), $\|\cdot\|$ is any norm, and $\lambda > 0$.

a. *[3 points]* Show that

$$\sum_{i=1}^{n} \ell_i(w) + \lambda \|w\|$$

is convex over $w \in \mathbb{R}^d$ (Hint: Show that if f, g are convex functions, then $f(x) + g(x)$ is also convex.)

b. *[1 points]* Explain in one sentence why we prefer to use loss functions and regularized loss functions that are convex.

What to Submit

- Proof for part a.
- Explanation for part b.

Lasso on a real dataset

Given $\lambda > 0$ and data $\Big(x_i, y_i)\Big)_{i=1}^{n}$, the Lasso is the problem of solving

$$\arg \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{i=1}^{n} (x_i^T w + b - y_i)^2 + \lambda \sum_{j=1}^{d} |w_j|$$

where λ is a regularization parameter. For the programming part of this homework, you will implement the coordinate descent method shown in Algorithm 1 to solve the Lasso problem.
You may use common computing packages (such as `numpy` or `scipy`), but do not use an existing Lasso solver (e.g., of `scikit-learn`).

Algorithm 1: Coordinate Descent Algorithm for Lasso

while *not converged* **do**
  $b \leftarrow \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{d} w_j x_{i,j}\Big)$
  for $k \in \{1, 2, \cdots d\}$ **do**
    $a_k \leftarrow 2 \sum_{i=1}^{n} x_{i,k}^2$
    $c_k \leftarrow 2 \sum_{i=1}^{n} x_{i,k} \Big(y_i - (b + \sum_{j \neq k} w_j x_{i,j})\Big)$
    $$w_k \leftarrow \begin{cases} (c_k + \lambda)/a_k & c_k < -\lambda \\ 0 & c_k \in [-\lambda, \lambda] \\ (c_k - \lambda)/a_k & c_k > \lambda \end{cases}$$
  end
end

Before you get started, the following hints may be useful:

- Wherever possible, use matrix libraries for matrix operations (not `for` loops).
- There are opportunities to considerably speed up parts of the algorithm by precomputing quantities like a_k before the for loop; you are permitted to add these improvements (and it may save you some time).
- As a sanity check, ensure the objective value is nonincreasing with each step.

- It is up to you to decide on a suitable stopping condition. A common criteria is to stop when no element of w changes by more than some small δ during an iteration. If you need your algorithm to run faster, an easy place to start is to loosen this condition.
- You will need to solve the Lasso on the same dataset for many values of λ. This is called a regularization path. One way to do this efficiently is to start at a large λ, and then for each consecutive solution, initialize the algorithm with the previous solution, decreasing λ by a constant ratio (e.g., by a factor of 2).
- The smallest value of λ for which the solution $\widehat{w}$ is entirely zero is given by

$$\lambda_{max} = \max_{k=1,\ldots,d} 2 \left| \sum_{i=1}^{n} x_{i,k} \left(y_i - \left(\frac{1}{n} \sum_{j=1}^{n} y_j \right) \right) \right| \tag{1}$$

  This is helpful for choosing the first λ in a regularization path.

A4. We will first try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting y, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features. Suppose that $x \in \mathbb{R}^d, y \in \mathbb{R}, k < d$, and data are generated independently according to the model $y_i = w^T x_i + \epsilon_i$ where

$$w_j = \begin{cases} j/k & \text{if } j \in \{1, \ldots, k\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is noise (note that in the model above $b = 0$). We can see from Equation (2) that since $k < d$ and $w_j = 0$ for $j > k$, the features $k+1$ through d are irrelevant for predicting y.
Generate a dataset using this model with $n = 500, d = 1000, k = 100$, and $\sigma = 1$. You should generate the dataset such that each $\epsilon_i \sim \mathcal{N}(0, 1)$, and y_i is generated as specified above.

a. *[10 points]* With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} where no features are selected (see Equation (1)) and decreasing λ by a constant ratio (e.g., 2) until nearly all the features are chosen. In plot 1, plot the number of non-zeros as a function of λ on the x-axis (Tip: use `plt.xscale('log')`).

b. *[10 points]* For each value of λ tried, record values for false discovery rate (FDR) (number of incorrect nonzeros in $\widehat{w}$/total number of nonzeros in $\widehat{w}$) and true positive rate (TPR) (number of correct nonzeros in $\widehat{w}$/k). Note: for each j, $\widehat{w}_j$ is an incorrect nonzero if and only if $\widehat{w}_j \neq 0$ while $w_j = 0$. In plot 2, plot these values with the x-axis as FDR, and the y-axis as TPR.

   Note that in an ideal situation we would have an (FDR,TPR) pair in the upper left corner. We can always trivially achieve $(0, 0)$ and $(\frac{d-k}{d}, 1)$.

c. *[5 points]* Comment on the effect of λ in these two plots in 1-2 sentences.

What to Submit:

- Your code implementing Algorithm 1 (submit to the HW2 code submission on Gradescope), and reproducing the experiments and plots for Plot 1 and Plot 2.
- Plot 1 described in part (a).
- Plot 2 described in part (b).
- 1-2 sentences for part (c).

A5. We'll now put the Lasso to work on some real data. Download the training data set "crime-train.txt" and the test data set "crime-test.txt" from the website. Store your data in your working directory, ensure you have the `pandas` library for Python installed, and read in the files with:

```
import pandas as pd
df_train = pd.read_table("crime-train.txt")
df_test = pd.read_table("crime-test.txt")
```

This stores the data as Pandas `DataFrame` objects. `DataFrame`s are similar to Numpy `array`s but more flexible; unlike `array`s, `DataFrame`s store row and column indices along with the values of the data. Each column of a `DataFrame` can also store data of a different type (here, all data are floats). Here are a few commands that will get you working with Pandas for this assignment:

```
df.head()                    # Print the first few lines of DataFrame df.
df.index                     # Get the row indices for df.
df.columns                   # Get the column indices.
df[``foo'''']                # Return the column named ``foo''''.
df.drop(``foo'', axis = 1)   # Return all columns except ``foo''.
df.values                    # Return the values as a Numpy array.
df[``foo''''].values         # Grab column foo and convert to Numpy array.
df.iloc[:3,:3]               # Use numerical indices (like Numpy) to get 3 rows and cols.
```

The data consist of local crime statistics for 1,994 US communities. The response y is the rate of violent crimes reported per capita in a community. The name of the response variable is `ViolentCrimesPerPop`, and it is held in the first column of `df_train` and `df_test`. There are 95 features. These features include many variables. Some features are the consequence of complex political processes, such as the size of the police force and other systemic and historical factors. Others are demographic characteristics of the community, including self-reported statistics about race, age, education, and employment drawn from Census reports.

The goals of this problem are threefold: (i) to encourage you to think about how data collection processes affect the resulting model trained from that data; (ii) to encourage you to think deeply about models you might train and how they might be misused; and (iii) to see how Lasso encourages sparsity of linear models in settings where d is large relative to n. **We emphasize that training a model on this dataset can suggest a degree of correlation between a community's demographics and the rate at which a community experiences and reports violent crime. We strongly encourage students to consider why these correlations may or may not hold more generally, whether correlations might result from a common cause, and what issues can result in misinterpreting what a model can explain.**

The dataset is split into a training and test set with 1,595 and 399 entries, respectively[1]. We will use this training set to fit a model to predict the crime rate in new communities and evaluate model performance on the test set. As there are a considerable number of input variables and fairly few training observations, overfitting is a serious issue. In order to avoid this, use the coordinate descent Lasso algorithm implemented in the previous problem.

a. *[4 points]* Read the documentation for the original version of this dataset: `http://archive.ics.uci.edu/ml/datasets/communities+and+crime`. Report 3 features included in this dataset for which historical *policy* choices in the US would lead to variability in these features. As an example, the *number of police* in a community is often the consequence of decisions made by governing bodies, elections, and amount of tax revenue available to decisionmakers.

b. *[4 points]* Before you train a model, describe 3 features in the dataset which might, if found to have nonzero weight in model, be interpreted as *reasons* for higher levels of violent crime, but which might actually be a *result* rather than (or in addition to being) the cause of this violence.

Now, we will run the Lasso solver. Begin with $\lambda = \lambda_{\max}$ defined in Equation (1). Initialize all weights to 0. Then, reduce λ by a factor of 2 and run again, but this time initialize $\hat{w}$ from your $\lambda = \lambda_{\max}$ solution as your initial weights, as described above. Continue the process of reducing λ by a factor of 2 until $\lambda < 0.01$. For all plots use a log-scale for the λ dimension (Tip: use `plt.xscale('log')`).

[1] The features have been standardized to have mean 0 and variance 1.

c. *[4 points]* Plot the number of nonzero weights of each solution as a function of λ.

d. *[4 points]* Plot the regularization paths (in one plot) for the coefficients for input variables `agePct12t29`, `pctWSocSec`, `pctUrban`, `agePct65up`, and `householdsize`.

e. *[4 points]* On one plot, plot the squared error on the training and test data as a function of λ.

f. *[4 points]* Sometimes a larger value of λ performs nearly as well as a smaller value, but a larger value will select fewer variables and perhaps be more interpretable. Inspect the weights $\hat{w}$ for $\lambda = 30$. Which feature had the largest (most positive) Lasso coefficient? What about the most negative? Discuss briefly.

g. *[4 points]* Suppose there was a large negative weight on `agePct65up` and upon seeing this result, a politician suggests policies that encourage people over the age of 65 to move to high crime areas in an effort to reduce crime. What is the (statistical) flaw in this line of reasoning? (Hint: fire trucks are often seen around burning buildings, do fire trucks cause fire?)

Logistic Regression

Binary Logistic Regression

A6. Here we consider the MNIST dataset, but for binary classification. Specifically, the task is to determine whether a digit is a 2 or 7. Here, let $Y = 1$ for all the "7" digits in the dataset, and use $Y = -1$ for "2". We will use regularized logistic regression. Given a binary classification dataset $\{(x_i, y_i)\}_{i=1}^n$ for $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ we showed in class that the regularized negative log likelihood objective function can be written as

$$J(w, b) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i(b + x_i^T w))) + \lambda ||w||_2^2$$

Note that the offset term b is not regularized. For all experiments, use $\lambda = 10^{-1}$. Let $\mu_i(w, b) = \frac{1}{1+\exp(-y_i(b+x_i^T w))}$.

a. *[8 points]* Derive the gradients $\nabla_w J(w, b)$, $\nabla_b J(w, b)$ and give your answers in terms of $\mu_i(w, b)$ (your answers should not contain exponentials).

b. *[8 points]* Implement gradient descent with an initial iterate of all zeros. Try several values of step sizes to find one that appears to make convergence on the training set as fast as possible. Run until you feel you are near to convergence.

   (i) For both the training set and the test, plot $J(w, b)$ as a function of the iteration number (and show both curves on the same plot).

   (ii) For both the training set and the test, classify the points according to the rule $\text{sign}(b + x_i^T w)$ and plot the misclassification error as a function of the iteration number (and show both curves on the same plot).

   Reminder: Make sure you are only using the test set for evaluation (not for training).

c. *[7 points]* Repeat (b) using stochastic gradient descent with a batch size of 1. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a). Show both plots described in (b) when using batch size 1. Take careful note of how to scale the regularizer.

d. *[7 points]* Repeat (b) using stochastic gradient descent with batch size of 100. That is, instead of approximating the gradient with a single example, use 100. Note, the expected gradient with respect to the random selection should be equal to the gradient found in part (a).

What to Submit

- Proof for part (a).
- Code which implements gradient descent, and reproduces your results in b, c, d.
- Separate plots for b(i) and b(ii).
- Separate plots for c which reproduce those from b(i) and b(ii) for this case.
- Separate plots for d which reproduce those from b(i) and b(ii) for this case.

B4.

Multinomial Logistic Regression *[25 points]*

We've talked a lot about binary classification, but what if we have $k > 2$ classes, like the 10 digits of MNIST? Concretely, suppose you have a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, k\}$ (i.e., k column vectors of length d, stacked side-by-side). Like in our least squares classifier of homework 1 for MNIST, we will assign a separate weight vector $\mathbf{w}^{(\ell)}$ for each class $\ell = 1, \ldots, k$; let $W = [\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}] \in \mathbb{R}^{d \times k}$. We can generalize the binary classification probabilistic model to multiple classes as follows: let

$$\mathbb{P}_W(y_i = \ell | W, \mathbf{x}_i) = \frac{\exp(\mathbf{w}^{(\ell)} \cdot \mathbf{x}_i)}{\sum_{j=1}^k \exp(\mathbf{w}^{(j)} \cdot \mathbf{x}_i)}$$

The negative log-likelihood function is equal to

$$\mathcal{L}(W) = -\sum_{i=1}^n \sum_{\ell=1}^k \mathbf{1}\{y_i = \ell\} \log \left(\frac{\exp(\mathbf{w}^{(\ell)} \cdot \mathbf{x}_i)}{\sum_{j=1}^k \exp(\mathbf{w}^{(j)} \cdot \mathbf{x}_i)} \right)$$

Define the $\texttt{softmax}(\cdot)$ operator to be the function that takes in a vector $\theta \in \mathbb{R}^k$ and outputs a vector in $\mathbb{R}^k$ whose mth component is equal to $\frac{\exp(\theta_m)}{\sum_{j=1}^k \exp(\theta_j)}$. Clearly, this vector is nonnegative and sums to one. If for any m we have $\theta_m \gg \max_{j \neq m} \theta_j$ then $\texttt{softmax}(\theta)$ approximates $\mathbf{e}_m$, a vector of all zeros with a one in the mth component.
For each y_i let $\mathbf{y}_i$ be the one-hot encoding of y_i (i.e., $\mathbf{y}_i \in \{0, 1\}^k$ is a vector of all zeros aside from a 1 in the y_ith index, or $\mathbf{e}_{y_i}$).

a. *[5 points]* If $\widehat{\mathbf{y}}_i^{(W)} = \texttt{softmax}(W^\top \mathbf{x}_i)$, show that $\nabla_W \mathcal{L}(W) = -\sum_{i=1}^n \mathbf{x}_i (\mathbf{y}_i - \widehat{\mathbf{y}}_i^{(W)})^\top$.

b. *[5 points]* Recall *Ridge Regression on MNIST* problem in Homework 1 and define $J(W) = \frac{1}{2} \sum_{i=1}^n \|\mathbf{y}_i - W^\top \mathbf{x}_i\|_2^2$. If $\widetilde{\mathbf{y}}_i^{(W)} = W^\top \mathbf{x}_i$, show that $\nabla_W J(W) = -\sum_{i=1}^n \mathbf{x}_i (\mathbf{y}_i - \widetilde{\mathbf{y}}_i^{(W)})^\top$. Remark: Comparing the least squares linear regression gradient step of this part to the gradient step of minimizing the negative log likelihood of the logistic model of part a may shed light on why we call this classification problem *logistic regression*.

c. *[15 points]* Using the original representations of the flattened MNIST images $\mathbf{x}_i \in \mathbb{R}^d$ ($d = 28 \times 28 = 764$) and all $k = 10$ classes, implement gradient descent (or stochastic gradient descent) for both $J(W)$ and $\mathcal{L}(W)$ and run until convergence on the training set. For each of the two solutions, report the classification accuracy on the training and test sets using the classification rule $\arg\max_j \mathbf{e}_j W^\top \mathbf{x}_i$.

   We highly encourage you to use `PyTorch` for this problem! The base object in `PyTorch` is the `torch.tensor`, which is essentially a `numpy.ndarray` but with some powerful new features. Namely, tensors have accelerator support (GPU, TPU) and high-performance autodifferentiation. Don't worry too much about the details of `PyTorch`! We will discuss `PyTorch` and the `torch.autograd` package in greater detail once we get to neural networks. For now, it is sufficient to know that `torch.autograd`

allows us to automatically calculate the gradients of our model parameters with minimal additional cost. Yep, that's right! Your days of writing out gradients by hand are numbered! :D

We include some starter pseudocode for the negative log-likelihood + softmax portion of this question. You are expected to find good hyperparameters. You can install the library at `https://pytorch.org/` and access the relevant beginner tutorial **here**.

```
import torch

W = torch.zeros(784, 10, requires_grad=True)
for epoch in range(epochs):
    y_hat = torch.matmul(X_train, W)
    # cross entropy combines softmax calculation with NLLLoss
    loss = torch.nn.functional.cross_entropy(y_hat, y_train)
    # computes derivatives of the loss with respect to W
    loss.backward()
    # gradient descent update
    W.data = W.data - step_size * W.grad
    # .backward() accumulates gradients into W.grad instead
    # of overwriting, so we need to zero out the weights
    W.grad.zero_()
```

What to Submit

(1) Proofs for parts a and b.

(2) Training and test classification accuracy for both $J(W)$ and $\mathcal{L}(W)$ from part c.

(3) Gradient descent implementations from part c (both in your PDF and in the Gradescope code assignment).

Confidence Interval of Least Squares Estimation

Bounding the Estimate *[30 points]*

B5. Let us consider the setting, where we have n inputs, $X_1, ..., X_n \in \mathbb{R}^d$, and n observations $Y_i = \langle X_i, \beta^* \rangle + \epsilon_i$, for $i = 1, ..., n$. Here, β^* is a ground truth vector in $\mathbb{R}^d$ that we are trying to estimate, the noise $\epsilon_i \sim \mathcal{N}(0, 1)$, and the n examples piled up — $X \in R^{n \times d}$. To estimate, we use the least squares estimator $\widehat{\beta} = \min_\beta \|X\beta - Y\|_2^2$. Moreover, we will use $n = 20000$ and $d = 10000$ in this problem.

a. *[3 points]* Show that $\widehat{\beta}_j \sim \mathcal{N}(\beta_j^*, (X^T X)^{-1}_{j,j})$ for each $j = 1, ..., d$. *(Hint: see notes on confidence intervals from lecture.)*

b. *[4 points]* Fix $\delta \in (0, 1)$ suppose $\beta^* = 0$. Applying the proposition from the notes, conclude that for each $j \in [d]$, with probability at least $1 - \delta$, $|\widehat{\beta}_j| \leq \sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$. Can we conclude that with probability at least $1 - \delta$, $|\widehat{\beta}_j| \leq \sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$ for all $j \in [d]$ simultaneously? Why or why not?

c. *[5 points]* Let's explore this question empirically. Assume data is generated as $x_i = \sqrt{(i \mod d) + 1} \cdot e_{(i \mod d)+1}$ where e_i is the ith canonical vector and $i \mod d$ is the remainder of i when divided by d. Generate each y_i according to the model above. Compute $\widehat{\beta}$ and plot each $\widehat{\beta}_j$ as a scatter plot with the x-axis as $j \in \{1, \ldots, d\}$. Plot $\pm\sqrt{2(X^T X)^{-1}_{j,j} \log(2/\delta)}$ as the upper and lower confidence

intervals with $1-\delta = 0.95$. How many $\widehat{\beta}_j$'s are outside the confidence interval? *Hint: Due to the special structure of how we generated x_i, we can compute $(X^TX)^{-1}$ analytically without computing an inverse explicitly.*

d. *[5 points]* For events $E_1, ..., E_m$ that are possibly dependent the "union bound" says that $P(\cup_i E_i) \leq \sum_i P(E_i)$. Use the union bound to come up with a threshold $\gamma > 0$ such that the probability that any of the coefficients $|\widehat{\beta}_j - \beta_j^*|$ exceed γ is less than δ.

e. *[7 points]* Fix $\mu > 0$. Assume each x_i is generated as above, and assume $\beta_i^* = \mu$ for $i \leq d/2$ and $\beta_i^* = 0$ for $i > d/2$. Using the threshold γ derived in part 3, find a sufficient condition for how big n needs to be to ensure that $\{j \in \{1, \ldots, d\} : |\widehat{\beta}_j| > \gamma\} = \{j \in \{1, \ldots, d\} : \beta_j^* \neq 0\}$ with probability at least $1 - \delta$.

f. *[6 points]* Is the union bound tight? Let $Z_1, \cdots, Z_d$ be i.i.d. random variables that follow $\mathcal{N}(0, 1)$. Show that there exist absolute constants c, C such that $c\sqrt{\log(d)} \leq \mathbb{E}[\max_{i=1,\cdots,d} Z_i] \leq C\sqrt{\log(d)}$. *Hint: Use the bounds on $\int_{x=t}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ given in the notes. For the lower bound use $E[Y] \geq t\mathbb{P}(Y \geq t)$. For the upper bound use the fact that $\mathbb{E}[Y] \leq \mathbb{E}[|Y|] = \int_{t=0}^{\infty} \mathbb{P}(|Y| \geq t) dt$.*

What to Submit:

- Mathematical proof for part (a).
- Answer and mathematical proof for part (b).
- Plots of $\hat{\beta}$ and its confidence interval **on the same plot** for part (c).
- Value of γ and how you mathematically derive it for part (d).
- The conditions and how you mathematically derive them for part (e).
- Mathematical proof for part (f).