

Homework #4

CSE 446: Machine Learning

Prof. Jamie Morgenstern

Due: **Friday** March 10, 2023 11:59pm

88 points

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 10% of the points for each problem improperly tagged.** For instructions, see https://www.gradescope.com/get_started#student-submission.
- If you collaborate on this homework with others, you must indicate who you worked with on your homework by providing a complete list of collaborators on the first page of your assignment. Make sure to include the name of each collaborator, and on which problem(s) you collaborated. Failure to do so may result in accusations of plagiarism. You can review the course collaboration policy at <https://courses.cs.washington.edu/courses/cse446/23wi/assignments/>
- **The coding problems are available in a .zip file on the course website**, with some starter code. All coding questions in this class will have starter code. **Before attempting these problems make sure your coding environment is working.** See instructions in README file in the .zip file.
- 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 adhering to these reminders may result in point deductions.

Conceptual Questions

A1. The answers to these questions should be answerable without referring to external materials. Briefly justify your answers with a few words.

- a. [2 points] True or False: Given a data matrix $X \in \mathbb{R}^{n \times d}$ where d is much smaller than n and $k = \text{rank}(X)$, if we project our data onto a k -dimensional subspace using PCA, our projection will have zero reconstruction error (in other words, we find a perfect representation of our data, with no information loss).
- b. [2 points] True or False: Suppose that an $n \times n$ matrix X has a singular value decomposition of USV^\top , where S is a diagonal $n \times n$ matrix. Then, the rows of V are equal to the eigenvectors of $X^\top X$.
- c. [2 points] True or False: choosing k to minimize the k -means objective (see Equation (1) below) is a good way to find meaningful clusters.
- d. [2 points] True or False: The singular value decomposition of a matrix is unique.
- e. [2 points] True or False: The rank of a square matrix equals the number of its unique nonzero eigenvalues.

What to Submit:

- **Parts a-e:** 1-2 sentence explanation containing your answer.

Think before you train

A2. **The first part of this problem (part a)** explores how you would apply machine learning theory and techniques to a real-world problem. There is one scenario detailing a setting, a dataset, and a specific result we hope to achieve. Your job is to describe how you would handle the scenario with the tools we've learned in this class. Your response should include:

- (1) any pre-processing steps you would take (i.e., data acquisition and processing),
- (2) the specific machine learning pipeline you would use (i.e., algorithms and techniques learned in this class),
- (3) how your setup acknowledges the constraints and achieves the desired result.

You should also aim to leverage some of the theory we have covered in this class. Some things to consider may be: the nature of the data (i.e., *How hard is it to learn? Do we need more data? Are the data sources good?*), the effectiveness of the pipeline (i.e., *How strong is the model when properly trained and tuned?*), and the time needed to effectively perform the pipeline.

a. *[10 points]* **Scenario: Disease Susceptibility Predictor**

- **Setting:** You are tasked by a research institute to create an algorithm that learns the factors that contribute most to acquiring a specific disease.
- **Dataset:** A rich dataset of personal demographic information, location information, risk factors, and whether a person has the disease or not.
- **Result:** The company wants a system that can determine how susceptible someone is to this disease when they enter in their own personal information. The pipeline should take limited amount of personal data from a new user and infer more detailed metrics about the person.

The second part of this problem (parts b, c) focuses on exploring possible shortcomings of machine learning models, and what real-world implications might follow from ignoring these issues.

- b. *[5 points]* Briefly describe (1) some potential shortcomings of your training process from the disease susceptibility predictor scenario above that may result in your algorithm having different accuracy on different populations, and (2) how you may modify your procedure to address these shortcomings.
- c. *[5 points]* Recall in Homework 2 we trained models to predict crime rates using various features. It is important to note that **datasets describing crime have many shortcomings in describing the entire landscape of illegal behavior in a city, and that these shortcomings often fall disproportionately on minority communities**. Some of these shortcomings include that crimes are reported at different rates in different neighborhoods, that police respond differently to the same crime reported or observed in different neighborhoods, and that police spend more time patrolling in some neighborhoods than others. What real-world implications might follow from ignoring these issues?

What to Submit:

- **For part (a):** One clearly-written short paragraph (4-7) sentences.
- **For part (b):** Clearly-written and well-thought-out answers addressing (1) and (2) (as described in the problem). Two short paragraphs or one medium paragraph suffice.
- **For part (c):** One clearly-written short paragraph on real-world implications that may follow from ignoring dataset issues.

Image Classification on CIFAR-10

A3. In this problem we will explore different deep learning architectures for image classification on the CIFAR-10 dataset. Make sure that you are familiar with `torch.Tensors`, two-dimensional convolutions (`nn.Conv2d`) and fully-connected layers (`nn.Linear`), ReLU non-linearities (`F.relu`), pooling (`nn.MaxPool2d`), and tensor reshaping (`view`).

A few preliminaries:

- Each network f maps an image $x^{\text{in}} \in \mathbb{R}^{32 \times 32 \times 3}$ (3 channels for RGB) to an output $f(x^{\text{in}}) = x^{\text{out}} \in \mathbb{R}^{10}$. The class label is predicted as $\arg \max_{i=0,1,\dots,9} x_i^{\text{out}}$. An error occurs if the predicted label differs from the true label for a given image.
- The network is trained via multiclass cross-entropy loss.
- Create a validation dataset by appropriately partitioning the train dataset. *Hint:* look at the documentation for `torch.utils.data.random_split`. Make sure to tune hyperparameters like network architecture and step size on the validation dataset. Do **NOT** validate your hyperparameters on the test dataset.
- At the end of each epoch (one pass over the training data), compute and print the training and validation classification accuracy.
- While one would usually train a network for hundreds of epochs to reach convergence and maximize accuracy, this can be prohibitively time-consuming, so feel free to train for just a dozen or so epochs.

For parts (a) and (b), apply a hyperparameter tuning method (e.g. random search, grid search, etc.) using the validation set, report the hyperparameter configurations you evaluated and the best set of hyperparameters from this set, and plot the training and validation classification accuracy as a function of epochs. Produce a separate line or plot for each hyperparameter configuration evaluated (top 3 configurations is sufficient to keep the plots clean). Finally, evaluate your best set of hyperparameters on the test data and report the test accuracy.

Note 1: Please refer to the provided notebook with starter code for this problem, included with the code files for this assignment. That notebook provides a complete end-to-end example of loading data, training a model using a simple network with a fully-connected output and no hidden layers (this is equivalent to logistic regression), and performing evaluation using canonical Pytorch. We recommend using this as a template for your implementations of the models below.

Note 2: If you are attempting this problem and do not have access to GPU we highly recommend using Google Colab. The provided notebook includes instructions on how to use GPU in Google Colab.

Here are the network architectures you will construct and compare.

- a. **[18 points] Fully-connected output, 1 fully-connected hidden layer:** this network has one hidden layer denoted as $x^{\text{hidden}} \in \mathbb{R}^M$ where M will be a hyperparameter you choose (M could be in the hundreds). The nonlinearity applied to the hidden layer will be the `relu` ($\text{relu}(x) = \max\{0, x\}$). This network can be written as

$$x^{\text{out}} = W_2 \text{relu}(W_1(x^{\text{in}}) + b_1) + b_2$$

where $W_1 \in \mathbb{R}^{M \times 3072}$, $b_1 \in \mathbb{R}^M$, $W_2 \in \mathbb{R}^{10 \times M}$, $b_2 \in \mathbb{R}^{10}$. Tune the different hyperparameters and train for a sufficient number of epochs to achieve a *validation accuracy* of at least 50%. Provide the hyperparameter configuration used to achieve this performance.

- b. **[18 points] Convolutional layer with max-pool and fully-connected output:** for a convolutional layer W_1 with filters of size $k \times k \times 3$, and M filters (reasonable choices are $M = 100$, $k = 5$), we have that $\text{Conv2d}(x^{\text{in}}, W_1) \in \mathbb{R}^{(33-k) \times (33-k) \times M}$.

- Each convolution will have its own offset applied to each of the output pixels of the convolution; we denote this as $\text{Conv2d}(x^{\text{in}}, W) + b_1$ where b_1 is parameterized in \mathbb{R}^M . Apply a `relu` activation to the result of the convolutional layer.

- Next, use a max-pool of size $N \times N$ (a reasonable choice is $N = 14$ to pool to 2×2 with $k = 5$) we have that $\text{MaxPool}(\text{relu}(\text{Conv2d}(x^{\text{in}}, W_1) + b_1)) \in \mathbb{R}^{\lfloor \frac{33-k}{N} \rfloor \times \lfloor \frac{33-k}{N} \rfloor \times M}$.
- We will then apply a fully-connected layer to the output to get a final network given as

$$x^{\text{output}} = W_2(\text{MaxPool}(\text{relu}(\text{Conv2d}(x^{\text{input}}, W_1) + b_1))) + b_2$$

where $W_2 \in \mathbb{R}^{10 \times M(\lfloor \frac{33-k}{N} \rfloor)^2}$, $b_2 \in \mathbb{R}^{10}$.

The parameters M, k, N (in addition to the step size and momentum) are all hyperparameters, but you can choose a reasonable value. Tune the different hyperparameters (number of convolutional filters, filter sizes, dimensionality of the fully-connected layers, step size, etc.) and train for a sufficient number of epochs to achieve a *validation accuracy* of at least 65%. Provide the hyperparameter configuration used to achieve this performance.

The number of hyperparameters to tune, combined with the slow training times, will hopefully give you a taste of how difficult it is to construct networks with good generalization performance. State-of-the-art networks can have dozens of layers, each with their own hyperparameters to tune. Additional hyperparameters you are welcome to play with if you are so inclined, include: changing the activation function, replace max-pool with average-pool, adding more convolutional or fully connected layers, and experimenting with batch normalization or dropout.

What to Submit:

- **For parts (a)-(b):** A single plot of the training and validation accuracy for the top 3 hyperparameter configurations you evaluated (x-axis is training epoch; y-axis is accuracy; this plot should contain 6 lines total). If it took fewer than 3 hyperparameter configurations to pass the performance threshold, plot all hyperparameter configurations you evaluated. A horizontal line should be plotted at the targeted threshold (50% or 65%). Validation lines should be dotted, and training lines should be solid.
- **For parts (a)-(b):** List the hyperparameter values you searched over and your search method (random, grid, etc.). Provide the values of best performing hyperparameters, and accuracy of best models on test data.
- **For parts (a)-(b):** Code. (**Note:** You do not need to submit the `.ipynb` notebook to the coding assignment on Gradescope. Please only submit your code for A3 in the PDF submission.)

k -means clustering

A4. Given a dataset $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and an integer $1 \leq k \leq n$, recall the following k -means objective function

$$\min_{\pi_1, \dots, \pi_k} \sum_{i=1}^k \sum_{j \in \pi_i} \|\mathbf{x}_j - \mu_i\|_2^2, \quad \mu_i = \frac{1}{|\pi_i|} \sum_{j \in \pi_i} \mathbf{x}_j. \quad (1)$$

Above, $\{\pi_i\}_{i=1}^k$ is a partition of $\{1, 2, \dots, n\}$. The objective (1) is NP-hard¹ to find a global minimizer of. Nevertheless, Lloyd's algorithm (discussed in lecture) typically works well in practice.

- a. [5 points] Implement Lloyd's algorithm for solving the k -means objective (1). Do not use any off-the-shelf implementations, such as those found in `scikit-learn`. Include your code in your submission.
- b. [5 points] Run Lloyd's algorithm on the *training* dataset of MNIST with $k = 10$. Show the image representing the center of each cluster, as a set of k 28×28 images.

Note on Time to Run — The runtime of a good implementation for this problem should be fairly fast (a few minutes); if you find it taking upwards of one hour, please check your implementation! (Hint: **For loops are costly**. Can you vectorize it or use Numpy operations to make it faster in some ways? If not, is looping through data-points or through centers faster?)

What to Submit:

- **For part (a):** Lloyd's algorithm code
- **For part (b):** 10 images of cluster centers.
- Code for parts a-b

¹To be more precise, it is both NP-hard in d when $k = 2$ and k when $d = 2$. See the references on the Wikipedia page for k -means for more details.

PCA

A5. Let's do PCA on MNIST dataset and reconstruct the digits in the dimensionality-reduced PCA basis. Compute your PCA basis using the training dataset only, and evaluate the quality of the basis on the test set, similar to the k-means reconstructions above. We have $n_{train} = 50,000$ training examples of size 28×28 . Begin by flattening each example to a vector to obtain $X_{train} \in \mathbb{R}^{50,000 \times d}$ and $X_{test} \in \mathbb{R}^{10,000 \times d}$ for $d = 784$.

Let $\mu \in \mathbb{R}^d$ denote the average of the training examples in X_{train} , i.e., $\mu = \frac{1}{n_{train}} X_{train}^\top \mathbf{1}$. Now let $\Sigma = (X_{train} - \mathbf{1}\mu^\top)^\top (X_{train} - \mathbf{1}\mu^\top) / 50000$ denote the sample covariance matrix of the training examples, and let $\Sigma = UDU^\top$ denote the eigenvalue decomposition of Σ .

- a. [2 points] If λ_i denotes the i th largest eigenvalue of Σ , what are the eigenvalues $\lambda_1, \lambda_2, \lambda_{10}, \lambda_{30}$, and λ_{50} ? What is the sum of eigenvalues $\sum_{i=1}^d \lambda_i$?
- b. [5 points] Let $x \in \mathbb{R}^d$ and $k \in 1, 2, \dots, d$. Write a formula for the rank- k PCA approximation of x .
- c. [3 points] Visualize a set of reconstructed digits from the training set for different values of k . In particular, provide the reconstructions for digits 2, 6, 7 with values $k = 5, 15, 40, 100$ (just choose an image from each digit arbitrarily). Show the original image side-by-side with its reconstruction. Provide a brief interpretation, in terms of your perceptions of the quality of these reconstructions and the dimensionality you used.

What to Submit:

- **For part (a):** Eigenvalues $\lambda_1, \lambda_2, \lambda_{10}, \lambda_{30}$, and λ_{50} and the sum. At least 6 leading digits.
- **For part (b):** The formula. If you are defining new variables/matrices, make sure their definition is stated clearly.
- **For part (c):** 15 total images, including 3 original and 12 reconstructed ones. Each reconstructed image corresponds to a certain digit (2, 6 or 7) and k value (5, 15, 40 or 100). In addition, provide your interpretation in a few sentences.
- **Code for parts a and c from `homeworks/pca/main.py`**

Administrative

A6.

- a. [2 points] About how many hours did you spend on this homework? There is no right or wrong answer :)