Homework #4

CSE 446/546: Machine Learning
Profs. Simon Du and Sewoong Oh
Due: **Friday** June 4, 2021 11:59pm Pacific Time

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 $\boxed{\text{boxed text}}$, 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 point]* .

Not adhering to these reminders may result in point deductions.

Changelog:

- **6/1:** Make A2b and A2c extra credit. Modify the wording for A3 and A4 to avoid confusion.
- **5/26:** Clarification on what to compare to for A4 part d. Title and clarity update for A6 description and subparts.
- **5/26:** More details on subgradients for A2.b.b.

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 $d \ll n$ 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 a matrix $X \in \mathbb{R}^{n\times n}$ 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 nonzero eigenvalues.

f. *[2 points]* True or False: Autoencoders, where the encoder and decoder functions are both neural networks with nonlinear activations, can capture more variance of the data in its encoded representation than PCA using the same number of dimensions.

What to Submit:

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

Basics of SVD and subgradients

A2.

a. Solve the following SVD problems

(a) *[3 points]* Let $\widehat{w}$ be the solution to the regression problem $\min_w \|Xw - y\|_2^2$. Let $\widehat{w}_{\text{R}}$ be the solution to the ridge regression problem $\min_w \|Xw - y\|_2^2 + \lambda \|w\|_2^2$. Let $X = U\Sigma V^\top$ be a singular value decomposition of X. Using this decomposition, explain why the solution $\widehat{w}_{\text{R}}$ to the ridge regression problem "shrinks" as compared to the solution $\widehat{w}$ of the standard regression problem.

(b) *[3 points]* Let $U \in \mathbb{R}^{n\times n}$ be a matrix with singular values all equal to one. Show that $UU^\top = U^\top U = I_n$. Further, use this result to show that U preserves Euclidean norms. In other words, $\|Ux\|_2 = \|x\|_2$ for any $x \in \mathbb{R}^n$.

b. *(extra credit)*

For a function f, its subgradient at $x \in \text{dom}(f)$ is the set $\{g\colon f(y) \geq f(x) + g^\top(y-x)\ \forall y \in \text{dom}(f)\}$. If $f = \max\{f_i\}_{i=1}^m$ is the pointwise maximum of convex subdifferentiable functions, then this implies that for each $x \in \text{dom}(f)$, there exists $i \in [m]$ such that $f(x) = f_i(x)$. Therefore if g is in the sub-gradienit set of f_i at x, then g is also in the sub-gradient set of f since for all $y \in \text{dom}(f)$

$$f(y) \geq f_i(y) \geq f_i(x) + g^\top(y-x) = f_i(x) + g^\top(y-x) = f(x) + g^\top(y-x)\ .$$

Use this to compute the subgradient set of the following functions for $x \in \mathbb{R}^n$:

(a) *[3 points]* $f(x) = \|x\|_1$

(b) *[3 points]* $f(x) = \max\{f_i(x)\}_{i=1}^m$, where f_i are all convex and continuously differentiable functions (i.e., gradients exist everywhere in the domain for f_i).
(Hint: If g_1 and g_2 are two subgradients, then any convex combination of g_1 and g_2 is also in the sub-gradient set.)

c. *(extra credit)*

[3 points] In this subproblem, you will need to use the result of part b above. Consider the function $f(x) = \max_{i=1,2,\ldots,n} |x_i - (1 + \eta/i)|$ for some constant η and for all $x \in \mathbb{R}^n$. Let v be any subgradient of f at any point in its domain. What is the best upper bound you can prove on $\|v\|_\infty$?

What to Submit:

- For part a.(a): An explanation either in English or math, that points out exactly where the difference is between regularized and non-regularized versions of linear regression.
- For part a.(b): A proof
- For parts b: Derivation of the solution.
- For part c: Derivation of the upper bound of $\|v\|_\infty$

PCA

A3. Let's do PCA on MNIST dataset and reconstruct the digits in the dimensionality-reduced PCA basis.

You will 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 of above. Because $n = 60,000$ training examples and $m = 10,000$ testing examples are of size 28×28 so begin by flattening each example to a vector to obtain $X_{\text{train}} \in \mathbb{R}^{n \times d}$ and $X_{\text{test}} \in \mathbb{R}^{m \times d}$ for $d784$.

Let $\mu \in \mathbb{R}^d$ denote the average of the training examples in X_{train}, i.e., $\mu = \frac{1}{n} X_{\text{train}}^\top \mathbf{1}^\top$. Now let $\Sigma = \left(X_{\text{train}} - \mathbf{1}\mu^\top\right)^\top (X_{\text{train}} - \mathbf{1}\mu^\top)/n$ denote the sample covariance matrix of the training examples.

a. *[2 points]* If λ_i denotes the i-th largest eigenvalue of Σ, what are the eigenvalues λ_1, λ_2, λ_{10}, λ_{30}, and λ_{50}? What is the sum of eigenvalues $\sum_{i=1}^d \lambda_i$?

b. *[5 points]* Any example $x \in \mathbb{R}^d$ (including those from either the training or test set) can be approximated using just μ and the first k eigenvalue, eigenvector pairs, for any $k = 1, 2, \ldots, d$. For any k, provide a formula for computing this approximation.

c. *[5 points]* Using this approximation, plot the reconstruction error from $k = 1$ to 100 (the X-axis is k and the Y-axis is the mean-squared error reconstruction error) on the training set and the test set (using the μ and the basis learned from the training set). On a separate plot, plot $1 - \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^d \lambda_i}$ from $k = 1$ to 100.

d. *[3 points]* Now let us get a sense of what the top PCA directions are capturing. Display the first 10 eigenvectors as images, and provide a brief interpretation of what you think they capture.

e. *[3 points]* Finally, 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 λ_1, λ_2, λ_{10}, λ_{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: Plot containing reconstruction error on train and test sets. Plot of $1 - \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^d \lambda_i}$.

- For part d: 10 eigenvectors as images.
- For part e: 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).
- Code for parts c-e

Unsupervised Learning with Autoencoders

A4. In this exercise, we will train two simple autoencoders to perform dimensionality reduction on MNIST. As discussed in lecture, autoencoders are a long-studied neural network architecture comprised of an encoder component to summarize the latent features of input data, and a decoder component to try and reconstruct the original data from the latent features.

Weight Initialization and PyTorch

We recommend using `nn.Linear` for your linear layers. You will not need to initialize the weights yourself; the default He/Kaiming uniform initialization in PyTorch will be sufficient. *Hint: we also recommend using the `nn.Sequential` module to organize your network class and simplify the process of writing the forward pass.*

Training

Use `optim.Adam` for this question. Experiment with different learning rates. Use mean squared error (`nn.MSELoss()` or `F.mse_loss()`) for the loss function and ReLU for the non-linearity in b.

a. *[10 points]* Use a network with a single linear layer. Let $W_{\mathrm{e}} \in \mathbb{R}^{h\times d}$ and $W_{\mathrm{d}} \in \mathbb{R}^{d\times h}$. Given some $x \in \mathbb{R}^d$, the forward pass is formulated as

$$\mathcal{F}_1(x) = W_{\mathrm{d}} W_{\mathrm{e}} x \ .$$

Run experiments for $h \in \{32, 64, 128\}$. For each of the different h values, report your final error and visualize a set of 10 reconstructed digits, side-by-side with the original image. *Note:* We omit the bias term in the formulation for notational convenience since `nn.Linear` learns bias parameters alongside weight parameters by default.

b. *[10 points]* Use a single-layer network with non-linearity. Let $W_{\mathrm{e}} \in \mathbb{R}^{h\times d}$, $W_{\mathrm{d}} \in \mathbb{R}^{d\times h}$, and activation $\sigma\colon \mathbb{R} \to \mathbb{R}$. Given some $x \in \mathbb{R}^d$, the forward pass is formulated as

$$\mathcal{F}_2(x) = \sigma\left(W_{\mathrm{d}} \sigma\left(W_{\mathrm{e}} x\right)\right) \ .$$

Report the same findings as asked for in part a (for $h \in \{32, 64, 128\}$.

c. *[5 points]* Now, evaluate $\mathcal{F}_1(x)$ and $\mathcal{F}_2(x)$ (use $h = 128$ here) on the test set and report the results.

d. *[5 points]* In a few sentences, compare the quality of the reconstructions from these two autoencoders with those of PCA from A3. You may want to re-run your code for PCA using the different h values as the number of top-k eigenvalues.

What to Submit:

- For parts a-b: Final training error and set of 10 reconstructed images of digits, side-by-side with the original image (10 images for each part).
- For part c: Errors of networks from part a and b on testing set.
- For part d: 2-3 sentences on differences in quality of solutions between PCA and Autoencoders, with example images.
- Code for parts a-c

k-means clustering

A5. Given a dataset $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and an integer $1 \leq k \leq n$, recall the following k-means objective function

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

Above, $\{\pi_i\}_{i=1}^k$ is a partition of $\{1, 2, \ldots, n\}$. The objective (1) is NP-hard[1] to find a global minimizer of. Nevertheless Lloyd's algorithm, the commonly-used heuristic which we 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 the algorithm on the *training* dataset of MNIST with $k = 10$, plotting the objective function (1) as a function of the iteration number. Visualize (and include in your report) the cluster centers as a 28×28 image.

c. *[5 points]* For $k = \{2, 4, 8, 16, 32, 64\}$ run the algorithm on the *training* dataset to obtain centers $\{\mu_i\}_{i=1}^k$. If $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $\{(\mathbf{x}'_i, y'_i)\}_{i=1}^m$ denote the training and test sets, respectively, plot the training error $\frac{1}{n}\sum_{i=1}^n \min_{j=1,\ldots,k} \|\mu_j - \mathbf{x}_i\|_2^2$ and test error $\frac{1}{m}\sum_{i=1}^m \min_{j=1,\ldots,k} \|\mu_j - \mathbf{x}'_i\|_2^2$ as a function of k on the same plot.

What to Submit:

- For part a: Llyod's algorithm code
- For part b: Plot of objective function. 10 images of cluster centers.
- For part c: Plot of objective function as function of c.
- Code for parts a-c

[1] To 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.

ML in the Real World

A6. In this question, you will explore how to apply machine learning theory and techniques to real-world problems. Each of the following statements details a setting, a dataset, and a specific result we hope to achieve. Your job is to briefly (4-7 sentences) describe how you would handle each of the below scenarios with the tools we've learned in this class.

Your response should include any **(1)** pre-processing steps you would take (i.e., data processing / acquisition), **(2)** the specific machine learning pipeline you would use (i.e., algorithms and techniques learned in this class), and **(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.

Because of the open-ended nature of the question, any thoroughly written responses will receive full credit.

a. *[5 points]* **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 personal information. The pipeline should take limited amount of personal data from a new user and infer more detailed metrics about the person.

b. *[5 points]* **Social Media App Facial Recognition Technology**

- Setting: You are tasked with developing a machine learning pipeline that can quickly map someone's face for the application of filters (i.e., Snapchat, Instagram).
- Dataset: A set of face images compiled from the company's employees and their families.
- Result: The company wants an algorithm that can quickly identify the key features of a person's face to apply a filter. (**Note:** Do not worry about describing the actual filter application).

c. *[5 points]* **Malware Detection:**

- Setting: You are tasked by a major tech company to create a service that ingests malware metadata and provides accurate assessments as to whether a new file is malicious or not.
- Dataset: A set of malware file metadata from users worldwide describing all attributes of offending files including its contents, publisher, and attributes of the resultant activity.
- Result: The company wants an accurate, scalable solution to detect whether incoming files are malicious for users across the globe.

What to Submit:

- For parts a-c: One short paragraph (4-7) sentences for each of the described scenarios.

Matrix Completion and Recommendation System

B1. You will build a personalized movie recommender system. We will use the 100K MovieLens dataset available at https://grouplens.org/datasets/movielens/100k/. There are $m = 1682$ movies and $n = 943$ users. Each user rated at least 20 movies, but some watched many more. The total dataset contains $100,000$ total ratings from all users. The goal is to recommend movies the users haven't seen.
Consider a matrix $R \in \mathbb{R}^{m \times n}$ where the entry $R_{i,j} \in \{1, \dots, 5\}$ represents the jth user's rating on movie i. A higher value represents that the user is more satisfied with the movie.

We may think of our historical data as some observed entries of this matrix while many remain unknown, and we wish to estimate the unknown ratings that each user would assign to each movie. We could use these ratings to recommend the "best" movies for each user.

The dataset contains user and movie metadata which we will ignore. We strictly use the ratings contained in the `u.data` file. Use this data file and the following python code to construct a training and test set:

```
import csv
import numpy as np
data = []
with open('u.data') as csvfile:
    spamreader = csv.reader(csvfile, delimiter='\t')
    for row in spamreader:
        data.append([int(row[0])-1, int(row[1])-1, int(row[2])])
data = np.array(data)

num_observations = len(data)   # num_observations = 100,000
num_users = max(data[:,0])+1   # num_users = 943, indexed 0,...,942
num_items = max(data[:,1])+1   # num_items = 1682 indexed 0,...,1681

np.random.seed(1)
num_train = int(0.8*num_observations)
perm = np.random.permutation(data.shape[0])
train = data[perm[0:num_train],:]
test = data[perm[num_train::],:]
```

The arrays `train` and `test` contain R_{train} and R_{test}, respectively. Each line takes the form "j, i, s", where j is the user index, i is the movie index, and s is the user's score.

Using `train`, you will train a model that can predict $\widehat{R} \in \mathbb{R}^{m \times n}$, how every user would rate every movie. You will evaluate your model based on the average squared error on `test`:

$$\mathcal{E}_{\text{test}}(\widehat{R}) = \frac{1}{|\text{test}|} \sum_{(i,j,R_{i,j}) \in \text{test}} (\widehat{R}_{i,j} - R_{i,j})^2.$$

Low-rank matrix factorization is a baseline method for personalized recommendation. It learns a vector representation $u_i \in \mathbb{R}^d$ for each movie and a vector representation $v_j \in \mathbb{R}^d$ for each user, such that the inner product $\langle u_i, v_j \rangle$ approximates the rating $R_{i,j}$. You will build a simple latent factor model.

You will implement multiple estimators and use the inner product $\langle u_i, v_j \rangle$ to predict if user j likes movie i in the test data. For simplicity, we will put aside best practices and choose hyperparameters by using those that minimize the test error. You may use fundamental operators from `numpy` or `pytorch` in this problem (`numpy.linalg.lstsq, SVD, autograd,` etc.) but not any precooked algorithm from a package like `scikit-learn`. If there is a question whether some package is not allowed for use in this problem, it probably is not appropriate.

a. *[5 points]* Our first estimator pools all users together and, for each movie, outputs as its prediction the average user rating of that movie in `train`. That is, if $\mu \in \mathbb{R}^m$ is a vector where μ_i is the average rating of the users that rated the ith movie, write this estimator $\widehat{R}$ as a rank-one matrix. Compute the estimate $\widehat{R}$. What is $\mathcal{E}_{\text{test}}(\widehat{R})$ for this estimate?

b. *[5 points]* Allocate a matrix $\widetilde{R}_{i,j} \in \mathbb{R}^{m\times n}$ and set its entries equal to the known values in the training set, and 0 otherwise. Let $\widehat{R}^{(d)}$ be the best rank-d approximation (in terms of squared error) approximation to $\widetilde{R}$. This is equivalent to computing the singular value decomposition (SVD) and using the top d singular values. This learns a lower-dimensional vector representation for users and movies, assuming that each user would give a rating of 0 to any movie they have not reviewed.

- For each $d = 1, 2, 5, 10, 20, 50$, compute the estimator $\widehat{R}^{(d)}$. We recommend using an efficient solver such as `scipy.sparse.linalg.svds`.
- Plot the average squared error of predictions on the training set and test set on a single plot, as a function of d.

Note that, in most applications, we would not actually allocate a full $m \times n$ matrix. We do so here only because our data is relatively small and it is instructive.

c. *[10 points]* Replacing all missing values by a constant may impose strong and potentially incorrect assumptions on the unobserved entries of R. A more reasonable choice is to minimize the MSE (mean squared error) only on rated movies. Define a loss function:

$$L\Big(\{u_i\}_{i=1}^m, \{v_j\}_{j=1}^n\Big) := \sum_{(i,j,R_{i,j})\in\text{train}} (\langle u_i, v_j\rangle - R_{i,j})^2 + \lambda \sum_{i=1}^m \|u_i\|_2^2 + \lambda \sum_{j=1}^n \|v_j\|_2^2 \tag{2}$$

where $\lambda > 0$ is the regularization coefficient. We will implement algorithms to learn vector representations by minimizing (2).

Since this is a non-convex optimization problem, the initial starting point and hyperparameters may affect the quality of $\widehat{R}$. You may need to tune λ and σ to optimize the loss you see.

- *Alternating minimization*: First, randomly initialize $\{u_i\}$ and $\{v_j\}$. Then, alternatate between (1) minimizing the loss function with respect to $\{u_i\}$ by treating $\{v_j\}$ as fixed; and (2) minimizing the loss function with respect to $\{v_j\}$ by treating $\{u_i\}$ as fixed. Repeat (1) and (2) until both $\{u_i\}$ and $\{v_j\}$ converge. Note that when one of $\{u_i\}$ or $\{v_j\}$ is given, minimizing the loss function with respect to the other part has a closed-form solution.
- Try $d \in \{1, 2, 5, 10, 20, 50\}$ and plot the mean squared error of train and test as a function of d.

Some hints: Common choices for initializing the vectors $\{u_i\}_{i=1}^m$, $\{v_j\}_{j=1}^n$ include: entries drawn from `np.random.rand()` scaled by some scale factor $\sigma > 0$ (σ is an additional hyperparameter), or using one of the solutions from part b or c. You should never be allocating an $m \times n$ matrix for this problem.

d. *[10 points]* Repeat part c, using batched SGD rather than alternating minimization.

Stochastic Gradient Descent: First, randomly initialize $\{u_i\}$ and $\{v_j\}$. Then take a batch of random samples (of size b from your training set and compute a gradient step, and repeat until convergence.

Note that batch size b, regularization constant λ, scaling parameter σ and learning rate η are all hyperparameters. Since this is a non-convex optimization, the results may be quite sensitive to these hyperparameters and to the initialization.

One strategy for choosing η is to select the largest constant value such that the loss L still tends to decrease. Another strategy is to pick a relatively large value of η and then scale it by some factor $\beta \in (0, 1)$ so that $\eta \mapsto \beta\eta$ every time a number of examples are seen that exceeds the size of the training set.

Feel free to modify the loss function to, say, different regularizers if it helps reduce the test error. See `http://www.optimization-online.org/DB_FILE/2011/04/3012.pdf` for some ideas.

e. *[5 points]* Briefly, in words, compare the results of the prior two parts. This is an example where the loss functions are identical, but the *algorithm* used has drastic impact on how much the model fits, overfits, or generalizes to new, unseen data.

f. (Extra credit: *[5 points]*). Implement any algorithm you'd like (you must implement it yourself; do not use an off-the-shelf algorithm from e.g. `scikit-learn`) to find an estimator that achieves a test error of no more than 0.9. Please include your code.

What to Submit:

- For parts a-d: Code implementation.
- For part a (In addition to code): Math formula for $\hat{R}$. Value for $\mathcal{E}_{test}(\hat{R})$.
- For part b (In addition to code): Plot of MSE on training and test set vs. d.
- For part c (In addition to code): Plot of MSE on training and test set vs. d.
- For part d (In addition to code): Plot of MSE on training and test set vs. d.
- For part e: 2-3 sentences describing difference between parts c and d.
- For part f: Error of the algorithm on test set. Any references to that algorithm (medium posts, papers, etc.). Code implementing new algorithm.