Homework #3

CSE 446/546: Machine Learning
Prof. Kevin Jamieson, Jamie Morgenstern
Due: **Wednesday** 11/25/2020 11:59 PM
A: 128 points/ B: 15 points

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

- Please provide succinct answers along with succinct reasoning for all your answers. Points may be deducted if long answers demonstrate a lack of clarity. Similarly, when discussing the experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. In other words, all your explanations, tables, and figures for any particular part of a question must be grouped together, which also includes **your code and plots**.
- 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 point deductions. 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 seperately. 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.

Conceptual Questions

A1. The following 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 $X \in R^{n \times d}$ 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 nonzero eigenvalues.

f. *[2 points]* Say you trained an SVM classifier with an RBF kernel ($K(u,v) = exp(-\frac{\|u-v\|_2^2}{2\sigma^2})$). If it underfits the training set, should you increase or decrease σ?

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 $\widehat{w}_R$ 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 $\widehat{w}_R$ 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 $UU^\top = U^\top U = I$. 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. Compute the subgradient set of the following functions.

   (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)

c. *[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. Let v be any subgradient of f at any point in its domain. What is the best upper bound you can prove on $\|v\|_\infty$?

Kernels and the Bootstrap

A3. *[5 points]* Suppose that our inputs x are one-dimensional and that our feature map is infinite-dimensional: $\phi(x)$ is a vector whose ith component is

$$\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i$$

for all nonnegative integers i. (Thus, ϕ is an infinite-dimensional vector.) Show that $K(x, x') = e^{-\frac{(x-x')^2}{2}}$ is a kernel function for this feature map, i.e.,

$$\phi(x) \cdot \phi(x') = e^{-\frac{(x-x')^2}{2}}.$$

Hint: Use the Taylor expansion of e^z. (This is the one dimensional version of the Gaussian (RBF) kernel).

A4. This problem will get you familiar with kernel ridge regression using the polynomial and RBF kernels. First, let's generate some data. Let $n = 30$ and $f_*(x) = 4\sin(\pi x)\cos(6\pi x^2)$. For $i = 1, \ldots, n$ let each x_i be drawn uniformly at random on $[0, 1]$ and $y_i = f_*(x_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, 1)$.
For any function f, the true error and the train error are respectively defined as

$$\mathcal{E}_{true}(f) = E_{XY}[(f(X) - Y)^2], \qquad \widehat{\mathcal{E}}_{train}(f) = \frac{1}{n}\sum_{i=1}^n (f(x_i) - y_i)^2.$$

Using kernel ridge regression, construct a predictor

$$\widehat{\alpha} = \arg\min_\alpha ||K\alpha - y||^2 + \lambda\alpha^T K\alpha \ , \qquad \widehat{f}(x) = \sum_{i=1}^n \widehat{\alpha}_i k(x_i, x)$$

where $K_{i,j} = k(x_i, x_j)$ is a kernel evaluation and λ is the regularization constant. Include any code you use for your experiments in your submission.

a. *[10 points]* Using leave-one-out cross validation, find a good λ and hyperparameter settings for the following kernels:

- $k_{poly}(x, z) = (1 + x^T z)^d$ where $d \in \mathbb{N}$ is a hyperparameter,
- $k_{rbf}(x, z) = \exp(-\gamma \|x - z\|^2)$ where $\gamma > 0$ is a hyperparameter[1].

Report the values of d, γ, and the λ values for both kernels.

b. *[10 points]* Let $\widehat{f}_{poly}(x)$ and $\widehat{f}_{rbf}(x)$ be the functions learned using the hyperparameters you found in part a. For a single plot per function $\widehat{f} \in \{\widehat{f}_{poly}(x), \widehat{f}_{rbf}(x)\}$, plot the original data $\{(x_i, y_i)\}_{i=1}^n$, the true $f(x)$, and $\widehat{f}(x)$ (i.e., define a fine grid on $[0, 1]$ to plot the functions).

c. *[5 points]* We wish to build bootstrap percentile confidence intervals for $\widehat{f}_{poly}(x)$ and $\widehat{f}_{rbf}(x)$ for all $x \in [0, 1]$ from part b.[2] Use the non-parametric bootstrap with $B = 300$ bootstrap iterations to find 5% and 95% percentiles at each point x on a fine grid over $[0, 1]$.

Specifically, for each bootstrap sample $b \in \{1, \ldots, B\}$, draw uniformly at randomly with replacement n samples from $\{(x_i, y_i)\}_{i=1}^n$, train an $\widehat{f}_b$ using the bth resampled dataset, compute $\widehat{f}_b(x)$ for each x in your fine grid; let the 5th percentile at point x be the largest value ν such that $\frac{1}{B}\sum_{b=1}^B \mathbf{1}\{\widehat{f}_b(x) \le \nu\} \le .05$, define the 95% analogously.

Plot the 5 and 95 percentile curves on the plots from part b.

d. *[5 points]* Repeat parts a, b, and c with $n = 300$, but use 10-fold CV instead of leave-one-out for part a.

e. *[5 points]* For this problem, use the $\widehat{f}_{poly}(x)$ and $\widehat{f}_{rbf}(x)$ learned in part d. Suppose $m = 1000$ additional samples $(x'_1, y'_1), \ldots, (x'_m, y'_m)$ are drawn i.i.d. the same way the first n samples were drawn.

Use the non-parametric bootstrap with $B = 300$ to construct a confidence interval on $\mathbb{E}[(Y - \widehat{f}_{poly}(X))^2 - (Y - \widehat{f}_{rbf}(X))^2]$ (i.e. randomly draw with replacement m samples denoted as $\{(\widetilde{x}'_i, \widetilde{y}'_i)\}_{i=1}^m$ from $\{(x'_i, y'_i)\}_{i=1}^m$ and compute $\frac{1}{m}\sum_{i=1}^m \left((\widetilde{y}'_i - \widehat{f}_{poly}(\widetilde{x}'_i))^2 - (\widetilde{y}'_i - \widehat{f}_{rbf}(\widetilde{x}'_i))^2\right)$, repeat this B times) and find 5% and 95% percentiles. Report these values.

Using this confidence interval, is there statistically significant evidence to suggest that one of $\widehat{f}_{rbf}$ and $\widehat{f}_{poly}$ is better than the other at predicting Y from X? (Hint: does the confidence interval contain 0?)

k-means clustering

A5. Given a dataset $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ and an integer $1 \le k \le n$, 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 $\{1, 2, ..., n\}$. The objective (1) is NP-hard[3] 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. *[3 points]* Show that for a set of points $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$, the point $\mathbf{y} \in \mathbb{R}^d$ that minimizes $\sum_{i=1}^n \|\mathbf{x}_i - \mathbf{y}\|_2^2$ is the centroid of the points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.

[1] Given a dataset $x_1, \ldots, x_n \in \mathbb{R}^d$, a heuristic for choosing a range of γ in the right ballpark is the inverse of the median of all $\binom{n}{2}$ squared distances $\|x_i - x_j\|_2^2$.

[2] See Hastie, Tibshirani, Friedman Ch. 8.2 for a review of the bootstrap procedure.

[3] 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.

b. *[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.

c. *[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.

d. *[5 points]* For values of $k = \{2, 4, 8, 16, 32, 64\}$, 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 can you conclude about k from your observations?

B1.

Intro to sample complexity

For $i = 1, \ldots, n$ let $(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ where $y_i \in \{-1, 1\}$ and x_i lives in some set $\mathcal{X}$ (x_i is not necessarily a vector). The 0/1 loss, or *risk*, for a deterministic classifier $f : \mathcal{X} \to \{-1, 1\}$ is defined as:

$$R(f) = \mathbb{E}_{XY}[\mathbf{1}(f(X) \neq Y)]$$

where $\mathbf{1}(\mathcal{E})$ is the indicator function for the event $\mathcal{E}$ (the function takes the value 1 if $\mathcal{E}$ occurs and 0 otherwise). The expectation is with respect to the underlying distribution P_{XY} on (X, Y). Unfortunately, we don't know P_{XY} exactly, but we do have our i.i.d. samples $\{(x_i, y_i)\}_{i=1}^n$ drawn from it. Define the *empirical risk* as

$$\widehat{R}_n(f) = \frac{1}{n}\sum_{i=1}^n \mathbf{1}(f(x_i) \neq y_i)$$

which is just an empirical estimate of our loss. Suppose that a learning algorithm computes the empirical risk $R_n(f)$ for all $f \in \mathcal{F}$ and outputs the prediction function $\widehat{f}$ which is the one with the smallest empirical risk. (In this problem, we are assuming that $\mathcal{F}$ is finite.) Suppose that the best-in-class function f^* (i.e., the one that minimizes the true 0/1 loss) is:

$$f^* = \arg\min_{f \in \mathcal{F}} R(f)\,.$$

a. *[2 points]* Suppose that for some $f \in \mathcal{F}$, we have $R(f) > \epsilon$. Show that $\mathbb{P}(\widehat{R}_n(f) = 0) \leq e^{-n\epsilon}$. (You may use the fact that $1 - \epsilon \leq e^{-\epsilon}$.)

b. *[2 points]* Use the *union bound* to show that

$$Pr(\exists f \in \mathcal{F} \text{ s.t. } R(f) > \epsilon \text{ and } \widehat{R}_n(f) = 0) \leq |\mathcal{F}|e^{-\epsilon n}.$$

Recall that the union bound says that if $A_1, \ldots, A_k$ are events in a probability space, then

$$Pr(A_1 \cup A_2 \cup \ldots \cup A_k) \leq \sum_{1 \leq i \leq k} Pr(A_i).$$

c. *[2 points]* Solve for the minimum ϵ such that $|\mathcal{F}|e^{-\epsilon n} \leq \delta$.

d. *[4 points]* Use this to show that with probability at least $1-\delta$

$$\widehat{R}_n(\widehat{f}) = 0 \quad \Longrightarrow \quad R(\widehat{f}) - R(f^*) \leq \frac{\log(|\mathcal{F}|/\delta)}{n}$$

where $\widehat{f} = \arg\min_{f \in \mathcal{F}} \widehat{R}_n(f)$.

Context: Note that among a larger number of functions $\mathcal{F}$ there is more likely to exist an $\widehat{f}$ such that $\widehat{R}_n(\widehat{f}) = 0$. However, this increased flexibility comes at the cost of a worse guarantee on the true error reflected in the larger $|\mathcal{F}|$. This tradeoff quantifies how we can choose function classes $\mathcal{F}$ that over fit. This sample complexity result is remarkable because it depends just on the number of functions in $\mathcal{F}$, not what they look like. This is among the simplest results among a rich literature known as statistical learning theory. Using a similar strategy, one can use Hoeffding's inequality to obtain a generalization bound when $\widehat{R}_n(\widehat{f}) \neq 0$.

B2.

Perceptron

One of the oldest algorithms used in machine learning (from early 60s) is an online algorithm for learning a linear threshold function called the Perceptron Algorithm:

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = 0$, and initialize t to 1. Also let's automatically scale all examples $\mathbf{x}$ to have (Euclidean) length 1, since this doesn't affect which side of the plane they are on.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

If we make a mistake on a positive $\mathbf{x}$ we get $\mathbf{w}_{t+1} \cdot \mathbf{x} = (\mathbf{w}_t + \mathbf{x}) \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} + 1$, and similarly if we make a mistake on a negative $\mathbf{x}$ we have $\mathbf{w}_{t+1} \cdot \mathbf{x} = (\mathbf{w}_t - \mathbf{x}) \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} - 1$. So, in both cases we move closer (by 1) to the value we wanted. Here is a link if you are interested in more details.

Now consider the linear decision boundary for classification (labels in $\{-1, 1\}$) of the form $\mathbf{w} \cdot \mathbf{x} = 0$ (i.e., no offset). Now consider the following loss function evaluated at a data point $(\mathbf{x}, y)$ which is a variant on the hinge loss.

$$\ell((\mathbf{x}, y), \mathbf{w}) = \max(0, -y(\mathbf{w} \cdot \mathbf{x})).$$

a. *[2 points]* Given a dataset of $(\mathbf{x}_i, y_i)$ pairs, write down a single step of subgradient descent with a step size of η if we are trying to minimize

$$\frac{1}{n} \sum_{i=1}^{n} \ell((\mathbf{x}_i, y_i), \mathbf{w})$$

for $\ell(\cdot)$ defined as above. That is, given a current iterate $\widetilde{\mathbf{w}}$ what is an expression for the next iterate?

b. *[2 points]* Use what you derived to argue that the Perceptron can be viewed as implementing SGD applied to the loss function just described (for what value of η)?

c. *[1 points]* Suppose your data was drawn iid and that there exists a $\mathbf{w}^*$ that separates the two classes perfectly. Provide an explanation for why hinge loss is generally preferred over the loss given above.

Neural Networks for MNIST

A6. In Homework 1, we used ridge regression for training a classifier for the MNIST data set. Students who did problem B.2 also used a random feature transform. In Homework 2, we used logistic regression to distinguish between the digits 2 and 7. Students who did problem B.4 extended this idea to multinomial logistic regression to distinguish between all 10 digits. In this problem, we will use PyTorch to build a simple neural network classifier for MNIST to further improve our accuracy.

We will implement two different architectures: a shallow but wide network, and a narrow but deeper network. For both architectures, we use d to refer to the number of input features (in MNIST, $d = 28^2 = 784$), h_i to refer to the dimension of the ith hidden layer and k for the number of target classes (in MNIST, $k = 10$). For the non-linear activation, use ReLU. Recall from lecture that

$$\text{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Weight Initialization

Consider a weight matrix $W \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$. Note that here m refers to the input dimension and n to the output dimension of the transformation $Wx + b$. Define $\alpha = \frac{1}{\sqrt{m}}$. Initialize all your weight matrices and biases according to $\text{Unif}(-\alpha, \alpha)$.

Training

For this assignment, use the Adam optimizer from `torch.optim`. Adam is a more advanced form of gradient descent that combines momentum and learning rate scaling. It often converges faster than regular gradient descent. You can use either Gradient Descent or any form of Stochastic Gradient Descent. Note that you are still using Adam, but might pass either the full data, a single datapoint or a batch of data to it. Use cross entropy for the loss function and ReLU for the non-linearity.

Implementing the Neural Networks

a. *[10 points]* Let $W_0 \in \mathbb{R}^{h \times d}$, $b_0 \in \mathbb{R}^h$, $W_1 \in \mathbb{R}^{k \times h}$, $b_1 \in \mathbb{R}^k$ and $\sigma(z) : \mathbb{R} \to \mathbb{R}$ some non-linear activation function. Given some $x \in \mathbb{R}^d$, the forward pass of the wide, shallow network can be formulated as:

$$\mathcal{F}_1(x) = W_1 \sigma(W_0 x + b_0) + b_1$$

Use $h = 64$ for the number of hidden units and choose an appropriate learning rate. Train the network until it reaches 99% accuracy on the training data and provide a training plot (loss vs. epoch). Finally evaluate the model on the test data and report both the accuracy and the loss.

b. *[10 points]* Let $W_0 \in \mathbb{R}^{h_0 \times d}$, $b_0 \in \mathbb{R}^{h_0}$, $W_1 \in \mathbb{R}^{h_1 \times h_0}$, $b_1 \in \mathbb{R}^{h_1}$, $W_2 \in \mathbb{R}^{k \times h_1}$, $b_2 \in \mathbb{R}^k$ and $\sigma(z) : \mathbb{R} \to \mathbb{R}$ some non-linear activation function. Given some $x \in \mathbb{R}^d$, the forward pass of the network can be formulated as:

$$\mathcal{F}_2(x) = W_2 \sigma(W_1 \sigma(W_0 x + b_0) + b_1) + b_2$$

Use $h_0 = h_1 = 32$ and perform the same steps as in part a.

c. *[5 points]* Compute the total number of parameters of each network and report them. Then compare the number of parameters as well as the test accuracies the networks achieved. Is one of the approaches (wide, shallow vs. narrow, deeper) better than the other? Give an intuition for why or why not.

Using PyTorch: For your solution, you may not use any functionality from the `torch.nn` module except for `torch.nn.functional.relu` and `torch.nn.functional.cross_entropy`. You must implement the networks $\mathcal{F}$ from scratch. For starter code and a tutorial on PyTorch refer to the section material here and B.4. on the previous homework.

PCA

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

You will actually 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 $50,000$ training examples are size 28×28 so 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 $d := 784$.

A7. Let $\mu \in \mathbb{R}^d$ denote the average of the training examples in X_{train}, i.e., $\mu = \frac{1}{n} X_{train}^\top \mathbf{1}^\top$. 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. (Please use the first 50000 images in the MNIST's training set as train and the last 10000 images in MNIST's training set as test.) Let $\Sigma = UDU^T$ denote the eigenvalue decomposition of Σ.

a. *[2 points]* If λ_i denotes the ith 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.