

## EEE598: Statistical Machine Learning: From Theory to Practice

Instructor: Dr. Lalitha Sankar

Homework Assignment #4

Total points: 100

Due Date and Time: 11/26/2022 11:59 PM

Student Name: \_\_\_\_\_

ASU Id: \_\_\_\_\_

### Instructions:


- (1) Every problem should begin on a new page (use the \newpage command).
- (2) Page numbers for each problem should be entered in Gradescope when uploading HW
- (3) Furthermore, **page numbers for every sub-problem within a problem should also be entered in Gradescope.**
- (4) Points are given for steps and work – show all work and steps and clarifications/arguments for the steps
- (5) Some rules on Python code files:
  - (i) Filename convention: For every problem, the corresponding Python filename should follow the following nomenclature:  $\langle \text{FirstInitialLastName-HWxx-Problem\#Subproblem\#} \rangle$ . For example, for HW3, problem 7a, my Python file should have the name: LSankar-HW3-7a.xxx
  - (ii) All subproblems can be in one file but the FILENAME has to be clarifying as in (i) above.
  - (iii) All Python files generated for an HW can be zipped and uploaded but again, if filenames are misleading then points will be deducted.
  - (iv) All code will also be checked for complete overlap with other student's code.
  - (v) **Every file should be independently executable.**
- (6) Enjoy!

Some instructions on how to submit solutions on Gradescope and Canvas:

- (1) **All plots have to be included in the PDF file uploaded to Gradescope.** It is not the TA's or grader's job to find it on the canvas submission. Plots not in the PDF submission on Gradescope will NOT graded.
- (2) **Plots without legend on what you are plotting them for will lead to points being deducted.** Keep in mind you have several plots and it is important for us to know what these plots correspond to.
- (3) **Canvas submission should be restricted to only CODE.**
- (4) **Please include a copy of your code in your Gradescope submission** – this can make grading easier for us as a first run (before we run your code using the canvas upload). This is particularly helpful if we find that your basic code is correct and your results are correct.

### Additional Notes:

- a. Please review all homework guidance above before submitting to Gradescope.
- b. Please provide succinct answers along with succinct reasoning for all your answers. Points may be deducted if long answers demonstrate a lack of clarity.
- c. 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.**
- d. **Late homework submissions will not be accepted after the deadline.**

$|x|$  

$$\partial f(x) = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \\ [-1, 1] & x = 0 \end{cases}$$

## Classification

1. Consider using a 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.

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

- (a) [10 points] Given a dataset of  $(\mathbf{x}_i, y_i)$  pairs, write down a single step of subgradient descent with a step size of  $\eta$  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  $\tilde{\mathbf{w}}$  what is an expression for the next iterate? [Hint: subgradient is treated in the section following gradient descent in the textbook by Shai Shalev-Schwartz and Shai Ben-David that we used to derive our proofs for GD and SGD.]

- (b) [5 points] Use what you derived to argue that the Perceptron (as formulated in class) can be viewed as implementing SGD applied to the loss function just described (for what value of  $\eta$ )?

- (c) [5 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. [Hint: Hinge loss given by  $\max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$  is the loss that results from abstracting SVM as an optimization problem when using linear separating planes. For this problem, choose  $\mathbf{w}$  and  $b$  appropriately.]

## Kernels

$$x_i \in \mathbb{R}$$

$$\phi(z_i) \in \mathbb{R}^\infty$$

2. [10 points] Suppose that our inputs are one dimensional and that our feature map is infinite dimensional:  $\phi(x)$  is a vector whose  $i$ th component is

$$\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i, \quad 0 \leq i \leq \infty$$

for all nonnegative integers  $i$ . (Thus,  $\phi$  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).

3. This problem will get you familiar with kernel ridge regression using the polynomial and RBF kernel. First let's generate some data. Let  $n = 30$  and  $f_*(x) = 4 \sin(\pi x) \cos(6\pi x^2)$ . For  $i = 1, \dots, n$ , 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 is defined as

$$\mathcal{E}_{true}(f) = E_{XY}[(f(X) - Y)^2]$$

whereas the training error is defined as

$$\hat{\mathcal{E}}_{train}(f) = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2.$$

Using kernel ridge regression, construct a predictor

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

where  $K_{i,j} = k(x_i, x_j)$  is a kernel evaluation and  $\lambda$  is the regularization constant.

$$= \langle \phi(x_i), \phi(x_j) \rangle$$

- (a) [10 points] Using leave-one-out cross validation, find a good  $\lambda$  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<sup>1</sup>.

$d = 1:50$   
 $\lambda = \text{powers of } 10$

Report the values of  $d$ ,  $\gamma$ , and the  $\lambda$  values for both kernels.

How to find  $\lambda$  → Do you hard code these?

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

- (c) [10 points] We wish to build Bootstrap percentile confidence intervals for  $\hat{f}_{poly}(x)$  and  $\hat{f}_{rbf}(x)$  for all  $x \in [0, 1]$  from part b. Use the non-parametric bootstrap with  $B = 300$  datasets to find the 5<sup>th</sup> and 95<sup>th</sup> percentiles at each  $x$  on a fine grid over  $[0, 1]$  (see Hastie, Tibshirani, Friedman Ch. 8.2 for a review). Specifically, for each dataset  $b \in \{1, \dots, B\}$ , draw uniformly at random with replacement  $n$  samples from  $\{(x_i, y_i)\}_{i=1}^n$ , train an  $\hat{f}_b$  using the  $b$ th resampled dataset, compute  $\hat{f}_b(x)$  for each  $x$  in your fine grid; let the 5<sup>th</sup> percentile at point  $x$  be the largest value  $\nu$  such that  $\frac{1}{B} \sum_{b=1}^B \mathbf{1}\{\hat{f}_b(x) \leq \nu\} \leq .05$ , define the 95<sup>th</sup> percentile analogously. Plot the 5 and 95 percentile curves on the plots from part b.

- (d) [10 points] Repeat all parts of this problem up to part 3.c with  $n = 300$  (you may just use 10-fold CV instead of leave-one-out)

- (e) [10 points] For this problem, use the  $\hat{f}_{poly}(x)$  and  $\hat{f}_{rbf}(x)$  learned in part d. Suppose  $m = 1000$  additional samples  $(x'_1, y'_1), \dots, (x'_m, y'_m)$  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  $\mathbb{E}[(Y - \hat{f}_{poly}(X))^2 - (Y - \hat{f}_{rbf}(X))^2]$  (i.e. randomly draw with replacement  $m$  samples denoted as  $\{(\tilde{x}'_i, \tilde{y}'_i)\}_{i=1}^m$  from  $\{(x'_i, y'_i)\}_{i=1}^m$  and compute  $\frac{1}{m} \sum_{i=1}^m ((\tilde{y}'_i - \hat{f}_{poly}(\tilde{x}'_i))^2 - (\tilde{y}'_i - \hat{f}_{rbf}(\tilde{x}'_i))^2)$ , repeat this  $B$  times) and find 5% and 95% percentiles. Using this confidence interval, is there statistically significant evidence to suggest that one of  $\hat{f}_{rbf}$  and  $\hat{f}_{poly}$  is better than the other at predicting  $Y$  from  $X$ ? (Hint: does the confidence interval contain 0?)

Context: One can compute means and variances and appeal to the Central Limit Theorem to compute confidence intervals. However, in almost all other settings (such as computing models and various related performance metrics) computing nearly-exact confidence intervals is impossible because there is information we simply do not have. The Bootstrap method gives us an ability to construct confidence intervals on nearly any statistical quantity of interest with minimal side-knowledge.

## k-means clustering

4. 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-hard<sup>2</sup> in terms of finding a global minimizer. Nevertheless the commonly used heuristic, known as Lloyd's algorithm, typically works well in practice. 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`.

- (a) [10 points] Run the algorithm on the training dataset of MNIST with  $k = 10$ , plotting the objective function (1) as a function of iteration. Visualize (and include in your report) the cluster centers as a  $28 \times 28$  image.

<sup>1</sup>Given a dataset  $x_1, \dots, x_n \in \mathbb{R}^d$ , a heuristic for choosing a range of  $\gamma$  in the right ballpark is the inverse of the median of all  $\binom{n}{2}$  squared distances  $\|x_i - x_j\|_2^2$ .

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

- (b) [10 points] For  $k = \{2, 5, 10, 20, 40, 80, 160, 320, 640, 1280\}$  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  $\frac{1}{n} \sum_{i=1}^n \min_{j=1, \dots, k} \|\mu_j - \mathbf{x}_i\|_2^2$  and test error  $\frac{1}{m} \sum_{i=1}^m \min_{j=1, \dots, k} \|\mu_j - \mathbf{x}'_i\|_2^2$  as a function of  $k$  on the same plot. (If the large values of  $k$  are taking unreasonably long to compute, try Agave cluster on ASU RC, or worst case, just go up to the largest value of  $k$  you can.)

## K means clustering Algorithm

①

Assignment.

initialise = np.select C n

for each point, d

$S_0$

=

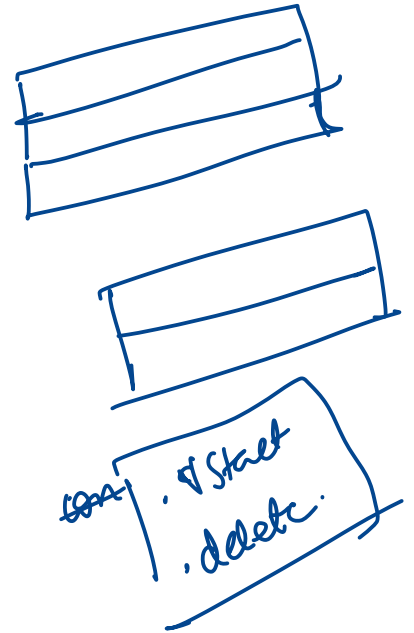
$S_1$

=

$S_2$

$\vdots$

$S_q$



new centroids  
old centroids