EEE598: Statistical Machine Learning: From Theory to Practice

Instructor: Dr. Lalitha Sankar
Homework Assignment #3
Total points: 100

Due Date and Time: 11/06/2022 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: ⟨FirstInitialLastName-HWxx-Problem#Subproblem#⟩. 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.**

Classification

1. *[10 points]* Suppose we run the Perceptron algorithm with $\mathbf{w}$ initialized to be an arbitrary unit vector. Suppose that the algorithm is then given the same vector $\mathbf{x}$ (whose label is 1) over and over again. How many mistakes can it make in the worst case? Express your answer in terms of $||\mathbf{x}||_2$. (Hint: derive the result from first principles, do not attempt to use the general result we discussed in class).

2. 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 $y_i \in \{1, \ldots, k\}$. 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 $\theta \in \mathbb{R}^{d \times k}$ and outputs a vector in $\mathbb{R}^k$ whose ith component is equal to $\frac{\exp(\theta_i)}{\sum_{j=1}^k \exp(\theta_j)}$. Clearly, this vector is non-negative and sums to one. If for any i we have $\theta_i \gg \max_{j \neq i} \theta_j$ then $\texttt{softmax}(\theta)$ approximates $\mathbf{e}_i$, a vector of all zeros with a one in the ith 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_i^{th} index).

(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 problem 6 of Homework 1 (*Ridge Regression on MNIST*) 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 $\nabla_W J(W) = -\sum_{i=1}^n \mathbf{x}_i (\mathbf{y}_i - \widetilde{\mathbf{y}}_i^{(W)})^\top$. 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 MNIST flattened images $\mathbf{x}_i \in \mathbb{R}^d$ ($d = 28 \times 28 = 784$) 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 of MNIST. For each of the two solutions, report the classification accuracy of each on the training and test sets using the most natural $\arg\max_j \mathbf{e}_j W^\top \mathbf{x}_i$ classification rule.

Multivariate Gaussians

3. For a matrix $A \in \mathbb{R}^{n \times n}$ we denote $|A|$ as the determinant of A. A multivariate Gaussian with mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{n \times n}$ has a probability density function $p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$ which we denote as $\mathcal{N}(\mu, \Sigma)$. For background on multivariate Gaussians, see Murphy 2.5.2, 2.6.1, and 4.1. Let

- $\mu_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$
- $\mu_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

For each $i = 1, 2$, on a separate plot:

(a) *[5 points]* Draw $n = 100$ points $X_{i,1}, \ldots, X_{i,n} \sim \mathcal{N}(\mu_i, \Sigma_i)$ and plot the points as a scatter plot with each point as a triangle marker (Hint: use `numpy.random.randn(d)` to generate a d-dimensional vector $Z \sim \mathcal{N}(0, I)$, then use the fact that $AZ + b \sim \mathcal{N}(b, AA^\top)$. Be careful, if symmetric A satisfies $A^2 = \Sigma$ then A is the matrix square root of Σ which in general is *not* the square root of the entries of Σ. See Murphy references above)

(b) *[10 points]* Compute the sample mean and covariance matrices $\widehat{\mu}_i = \frac{1}{n}\sum_{j=1}^{n} X_{i,j}$ and $\widehat{\Sigma}_i = \frac{1}{n-1}\sum_{j=1}^{n}(X_{i,j} - \widehat{\mu}_i)(X_{i,j} - \widehat{\mu}_i)^\top$. Compute the eigenvectors of $\widehat{\Sigma}_i$. Plot the unit-norm eigenvectors as line segments originating from $\widehat{\mu}_i$ and have magnitude equal to the square root of their corresponding eigenvalues. Make sure your axes are square (e.g., the x and y limits are both $[-r, r]$ for some appropriately large $r > 0$ to see the data.) (Hint: see `numpy.linalg.eig`.)

(c) *[10 points]* If $(u_{i,1}, \lambda_{i,1})$ and $(u_{i,2}, \lambda_{i,2})$ are the eigenvector-eigenvalue pairs of the sample covariance matrix with $\lambda_{i,1} \geq \lambda_{i,2}$ and $||u_{i,1}||_2 = ||u_{i,2}||_2 = 1$, for $j = 1, \ldots, n$ let $\widetilde{X}_{i,j} = \begin{bmatrix} \frac{1}{\sqrt{\lambda_{i,1}}} u_{i,1}^T (X_{i,j} - \widehat{\mu}_i) \\ \frac{1}{\sqrt{\lambda_{i,2}}} u_{i,2}^T (X_{i,j} - \widehat{\mu}_i) \end{bmatrix}$. On a new figure, plot these new points as a scatter plot with each point as a circle marker. Make sure your axes are square.

Classifying Gaussian Mixtures

4. Now consider a setting where the Gaussian distributions P_{-1} and P_1 for the feature vector $\mathbf{X} = [X_1 \; X_2]^T$ for classes $Y = -1$ and $Y = 1$, respectively, have parameters:

- $\mu_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$
- $\mu_{-1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\Sigma_{-1} = \Sigma_1$

Assume a uniform prior on the classes, i.e., $P_Y[-1] = P_Y[-1] = 1/2$. Answer the following questions.

(a) *[10 points]* Mirroring the method in class for the 1-D feature case, derive the optimal Bayes classification rule $\mathbf{w}^* = [w_1^* w_2^*]^T$ for this two-dimensional setting.

(b) *[8 points]* Using a 3-d plot, plot both the distributions (that are functions of the two features x_1 and x_2) as well as the two-dimensional Bayes optimal separating plane, namely $\sum_{i=1}^{2} x_i w_i^*$. Points are given for clearly delineating the distributions, planes, labeling axes, etc. [HINT: Create a grid of (x_1, x_2) values first to plot the distributions; use the same grid to compute the inner product of the resulting $\mathbf{x}$ vector and the optimal $\mathbf{w}^*$. Note that going up to 4 or 5 times the standard deviation should capture significant area under the curve; you will use the same grid for the plots required below].

(c) *[12 points]* Now generate 5,000 examples of (x_i, y_i) for each class, using your solution in problem 3.a for generating the x_i. Using a logistic regression model, i.e., $\hat{P}(y|x) = \sigma(\mathbf{w}^T \mathbf{x})$ where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{w} \in \mathbb{R}^2$ is the model weight, and $\mathbf{x} \in \mathbb{R}^2$ is a feature vector, learn the empirical risk minimization (ERM) logistic classifier $\mathbf{w}_{\text{ERM}}$ using gradient descent for this data set. In other words, use the methods we developed in class for learning a logistic classifier. **Repeat this experiment a 100 times using the same data (but different seeds, and hence, different starting vectors for the weights) to learn an averaged model $\bar{\mathbf{w}} = [\bar{w}_1 \bar{w}_2]^T$** by averaging the models $\mathbf{w}_{\text{ERM}}$ learned over 100 runs. Plot the two-dimensional averaged model you learn, i.e., $\sum_{k=1}^{2} x_k \bar{w}_k$ line; compare it with the optimal line $\sum_{k=1}^{2} x_k w_k^*$ learned in problem 4.a by plotting that as well.

(d) *[10 points]* Now generate 1000 samples from each class. Using the hard-decision rule of assigning $Y = 1$ for $(\hat{P}(y|x) > 1/2)$ (resp. $Y = -1$ for $(\hat{P}(y|x) < 1/2)$), and arbitrary (fair coin flip, for example) assignment of class for $(\hat{P}(y|x) = 1/2)$, compute the accuracy (ratio of the total number of correct classifications to the total number of samples) of the ERM logistic classifier. Compare that with the accuracy you obtain using the Bayesian model learned in 4.a for this same validation dataset.