CS 446 / ECE 449 — Homework 2

your NetID here

Version 1.0

Instructions.

- Homework is due **Wednesday, October 1^{st}, 11:59 a.m**; you have **3** late days in total for **all Homeworks**.
- Everyone must submit **individually** at gradescope under `hw2` and `hw2code`.
- The "written" submission at `hw2` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw2`, gradescope will ask you to **mark out boxes around each of your answers**; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full **academic integrity** information. You should cite any external reference you use.
- We reserve the right to reduce the auto-graded score for `hw2code` if we detect funny business (e.g., your solution lacks any algorithm and hard-code answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `HW2 coding`, only upload `hw2_q2.py`, `hw2_q4.py` and `hw2_utils.py`. Additional files will be ignored. (**DO NOT change the name of these three files!**)

1. Naive Bayes (25 pt)

(a) In Naive Bayes classification, the number of parameters to be estimated for a Bayesian classifier is reduced by assuming conditional independence when modeling $P(X|Y)$. Conditional independence is defined as follows:

Definition: Let X, Y, and Z be random variables. We say that X is conditionally independent of Y given Z, written as $X \perp Y|Z$, if and only if:

$$P(X = x_i|Y = y_j, Z = z_k) = P(X = x_i|Z = z_k), \forall i, j, k$$

Given this definition, please answer the following questions:

i. If $X \perp Y|Z$, can we conclude that $P(X, Y|Z) = P(X|Z)P(Y|Z)$? Explain your reasoning. (2 pt)

ii. If $X \perp Y|Z$, can we conclude that $P(X, Y) = P(X)P(Y)$? Explain your reasoning. (2 pt)

iii. Suppose X is a vector of d Boolean attributes, and Y is a discrete variable that takes C possible values. Let $\theta_{jc} = P(X_j|Y = c)$. How many independent θ_{jc} parameters must be estimated? (2 pt)

iv. Now suppose X is a vector of d real-valued attributes, and each X_j follows a Normal (Gaussian) distribution: $P(X_j = x_j|Y = c) = \mathcal{N}(x_j|\mu_{jc}, \sigma_{jc})$. How many distinct μ_{jc} and σ_{jc} parameters must be estimated? (2 pt)

v. We can write the classification rule for Naive Bayes as:

$$Y^{new} \leftarrow \arg\max_c \frac{P(Y = c)\prod_{j=1}^{d} P(X_j^{new}|Y = c)}{\sum_{v=1}^{C} P(Y = v)\prod_{k=1}^{d} P(X_k^{new}|Y = v)}$$

When estimating Y, we often omit the denominator in the calculation. Why is it acceptable to do this? (2 pt)

vi. Is it possible to compute $P(X)$ using the parameters estimated in Naive Bayes? (2 pt)

(b) Consider a classification problem where the input vector $X =< X_1, X_2, X_3 >$ consists of three boolean features $X_j \in \{0, 1\}, j = \{1, 2, 3\}$ and the label $Y \in \{0, 1\}$. You are given a dataset of N i.i.d. labeled examples $\{X^{(i)}, y^{(i)}\}_{i=1}^{N}$. For X_1, X_2 and X_3 we have: $P(X_1|X_2, X_3, Y) = P(X_1|Y)$, $P(X_2|X_1, X_3, Y) = P(X_2|Y)$ and $P(X_3|X_1, X_2, Y) = P(X_3|X_1)$.

i. Express the joint distribution $P(X_1, X_2, X_3, Y)$ as a product of simpler conditional probabilities, i.e. each variable depends only on atmost one another variable. (2 pt)

ii. Derive the maximum likelihood estimators for the following quantities (6 pt, 2pt each) :

A. $P(Y = 1)$

B. $P(X_1 = 1 \mid Y = y)$, for $y \in \{0, 1\}$.

C. $P(X_3 = 1 \mid Y = y)$, for $y \in \{0, 1\}$.

Note: It is sufficient to leave your answers as sums of indicator functions or fractions.

(c) Consider the same conditional independence structure as in Part (b), but this time with continuous features X_j drawn from Gaussian distributions instead of boolean. Specifically, let $P(Y = 0) = P(Y = 1) = 0.5$ and

$$\begin{aligned}
&(X_1 \mid Y = y) \sim \mathcal{N}(\mu_{1y}, 1), && \mu_{10} = 0,\ \mu_{11} = 1,\\
&(X_2 \mid Y = y) \sim \mathcal{N}(\mu_{2y}, 1), && \mu_{20} = 0,\ \mu_{21} = 1,\\
&(X_3 \mid X_1 = x_1) \sim \mathcal{N}(2x_1, 1) && \text{(independent of } Y \text{ given } X_1).
\end{aligned}$$

i. Derive the MAP rule for a new point X^{new}. (2 pt)

ii. Using your classification rule, classify the following two points:

$$X^{(a)} =< 0.2, 0.7, -10 >, \qquad X^{(b)} =< 0.2, 0.7, 10 > .$$

Do the predicted labels differ between these two cases? Explain why or why not. (3 pt)

Your solution here.

2. Gaussian Naive Bayes. (25 pt)

Recall from Lecture 7 (slide 5), taking $\log(\cdot)$ of the objective function, we have the decision rule as :

$$Y^{\text{new}} = \arg\max_y \left(\left(\sum_{j=1}^{d} \log P(X_j^{\text{new}} | Y = y) \right) + \log P(Y = y) \right)$$

In a Gaussian Naive Bayes, the features X_j^{new} are continuous variables, and the probability $P(X_j^{\text{new}} | Y = y)$ is modeled as a Gaussian distribution

$$P(X_j^{\text{new}} | Y = y) = \mathcal{N}(X_j^{\text{new}} \mid \mu_{y,j}, \sigma_{y,j}^2)$$

For simplicity, we assume that there exists at least example that belong to each class.

- There are d attributes to describe each example.
- We use pair $(\boldsymbol{x}^{(i)}, y^{(i)})$ to represent ith example, where $\boldsymbol{x}^{(i)}$ is a length-d vector that describes its properties, and $y^{(i)}$ is a scalar representing its label.
- For $j \in \{1, 2, \cdots, d\}$, $x_j^{(i)} \in \mathbb{R}$ is the value of attribute j of each example i; $y^{(i)} = 0$ means example i is in class 0, and $y^{(i)} = 1$ means example i is in class 1.
- We use $(\boldsymbol{X}, \boldsymbol{y})$ to represent a dataset of size N, where $\boldsymbol{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)} \right)^\top$ is a $N \times d$ matrix, and $\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, \cdots, y^{(N)} \right)$ is a length-N vector.

You are given two datasets in this homework: $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ and $(\boldsymbol{X}_{\text{test}}, \boldsymbol{y}_{\text{test}})$, which can be obtained by calling `gaussian_dataset("train")` and `gaussian_dataset("test")` in `hw2_utils.py`. Your tasks are:

(a) Implement the function `gaussian_theta(X, y)` in `hw2_q2.py`.
The input is the training dataset $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$. The output is $(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$. Both of them are $2 \times d$ matrices in PyTorch float tensor, where $\mu_{y,j}$ and $\sigma_{y,j}^2$ are Gaussian distribution parameters of the MLE estimation of $P(X_j = 1 \mid Y = y)$. (8 pt)

(b) Implement the function `gaussian_p(y)` in `hw2_q2.py`.
The input is the label part $\boldsymbol{y}_{\text{train}}$ of the training dataset. The output p is a scalar, which is the MLE for $P(Y = 0)$. (4 pt)

(c) Implement the function `gaussian_classify(mu, sigma2, p, X)` in `hw2_q2.py`.
The input is the output $\boldsymbol{\mu}, \boldsymbol{\sigma^2}, p$ from the two functions above, and the label part $\boldsymbol{X}_{\text{test}}$ of the testing dataset of size N. The output $\widehat{\boldsymbol{y}}$ is an length-N vector, where $\widehat{\boldsymbol{y}}^{(i)}$ is the predicted label (0 or 1) for the object with properties $\boldsymbol{x}_{\text{test}}^{(i)}$, or the i-th row of $\boldsymbol{X}_{\text{test}}$. For simplicity, it's guaranteed that there will be no ties (so the arg max will be unique). (6pt)
Note: Please use the logarithmic form of both $\widehat{Y}$ and Gaussian PDF to avoid precision issues.
Library routines: `torch.log`, `torch.sum`, `torch.mean`, `torch.var` (Please use `unbiased=False`).

(d) Show that, in Gaussian Naive Bayes with equal class priors $P(Y = 0) = P(Y = 1) = 0.5$ and with class-conditional distributions

$$(X_j \mid Y = y) \sim \mathcal{N}(\mu_{y,j}, \sigma_j^2),$$

where the variances σ_j^2 are the same across classes (but may differ across features), the MAP classification rule is equivalent to a *linear classifier* in $\boldsymbol{x} =< x_1, \ldots, x_d >$. That is, derive the explicit form of the decision boundary and show it can be written as

$$\widehat{y}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \sum_{j=1}^{d} w_j x_j > \tau, \\ 0, & \text{if } \sum_{j=1}^{d} w_j x_j < \tau. \end{cases}$$

for some weights w_j and threshold τ depending on $(\mu_{0,j}, \mu_{1,j}, \sigma_j^2)$. (7pt)

Your solution here.

3. Logistic Regression. (25 pt)

In logistic regression, the class probability is modeled as

$$P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \sigma(y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)})$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^{d+1}$ represents the feature vector, $y^{(i)} \in \{-1, +1\}$ is the class label, and $\sigma(s) = \frac{1}{1+e^{-s}}$ is the sigmoid function.

(Note: We exclude the bias term w_0 as it can be absorbed into the weight vector: $\mathbf{w} = \begin{bmatrix} \uparrow \\ \boldsymbol{w} \\ \downarrow \\ w_0 \end{bmatrix}$, and transform $\boldsymbol{x}$ to $\mathbf{x} = \begin{bmatrix} \uparrow \\ \boldsymbol{x} \\ \downarrow \\ 1 \end{bmatrix}$, which we discussed in Lecture 3 "notation hack".)

(a) Prove that the sigmoid function $\sigma(\cdot)$ satisfies the property

$$\sigma(-s) = 1 - \sigma(s)$$

By demonstrating this, show that, $P(y^{(i)} = -1|\boldsymbol{x}^{(i)}) + P(y^{(i)} = 1|\boldsymbol{x}^{(i)}) = 1$. (2 pt)

(b) Prove that

$$\sigma'(s) = \sigma(s)(1 - \sigma(s))$$

(2 pt)

(c) Derive the gradient of the log-likelihood function, that is, compute

$$\nabla_w \log P(\boldsymbol{y}|\mathbf{X}, \mathbf{w})$$

where $\boldsymbol{y} = [y^{(1)}, y^{(2)}, ..., y^{(N)}]^T$ and $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$. (5 pt)

(d) Derive the Hessian matrix $\boldsymbol{H}$ of the log-likelihood function, where each entry H_{ab} is given by

$$H_{ab} = \frac{\partial^2}{\partial w_a \partial w_b} \log P(\boldsymbol{y}|\mathbf{X}, \mathbf{w})$$

(5 pt)

(e) Prove that the Hessian is negative semi-definite, i.e., show that $\boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} \leq 0$ for any vector $\boldsymbol{z} \in \mathbb{R}^{d+1}$. By doing so, we can conclude that the log-likelihood is concave and has no local maxima, only a global maximum. (5 pt)

(f) **Gradient Descent Update Rule:** Use the gradient of the log-likelihood to derive the weight update rule for one iteration of gradient descent. Assume a learning rate α. (3 pt)

(g) **Newton's Method Update Rule:** Use Newton's method incorporating the Hessian matrix to derive the weight update rule. (3 pt)

Your solution here.

4. Programming - Optimization. (25 pt)

This assignment guides you through building, refining, and optimizing a Logistic Regression model from scratch. Complete the `TODO` sections in the provided Python code. You can either use the Jupyter Notebook (hw2_q4.ipynb) or the Python file (hw2_q4.py).

Submission Requirements: If you use the Jupyter notebook, please make sure to convert it to hw2_q4.py and submit it to Gradescope along with hw2_q2.py and hw2_utils.py. Submit all generated plots and write the discussion sections (4(a)iii, 4(b)ii, 4(d)ii) with your written assignment.

(a) Logistic Regression

i. Start by implementing the core components of a logistic regression model. (4 pt)

A. Implement the **sigmoid function**.

B. Implement the **cost function** (binary cross-entropy). A cost function measures the total error between a model's predictions and the actual labels, summarizing its performance into a single number to be minimized. Binary cross-entropy (or Log Loss) is a specific cost function for binary classification that heavily penalizes predictions that are both confident and incorrect. Please see code for more information on how to implement this.

C. Compute the **gradients** for the weights and bias.

D. Implement the **parameter update rule** for gradient descent.

ii. **Feature Transformation:** Create new features from the existing ones using non-linear transforms (e.g., x_1^2, x_2^2, x_1x_2) to map the data into a higher-dimensional space. (4 pt)

iii. **Discussion:** Analyze the generated plot of the decision boundary and explain why the model with transformed features can now correctly separate the data, whereas a linear model in the original feature space could not. (2 pt)

(b) L1 vs. L2 Regularization

i. For the same model, modify the cost function and gradient update steps to include options for L1 and L2 regularization. Note that L1 regularization (Lasso) adds a penalty equal to the sum of the absolute values of the weights. In contrast, L2 regularization (Ridge) adds a penalty equal to the sum of the squares of the weights. (4 pt)

ii. **Discussion:** Train three separate models: one with L1 regularization, one with L2 regularization, and one with no regularization. Generate a plot that compares their decision boundaries. Compare the effects of L1 and L2 regularization on the model. What kind of effect does regularization have on the model weights? What is the main difference in how they affect the model's weights? When would you choose one over the other? (2 pt)

(c) Hyperparameter Tuning (2 pt)

i. This exercise will guide you through performing a grid search to tune the learning rate (`alpha`) and the L2 regularization (`lambda`) hyperparameters.

A. Define a set of `alpha` values and `lambda` values to test.

B. Train a model for each combination of `alpha` and `lambda`, and evaluate its performance on the validation set.

ii. Identify the best-performing hyperparameter combination. Train a final model on the full training set using these optimal values and generate a plot of its decision boundary.

(d) Gradient Descent Variants

i. Run through the implementation of Batch Gradient Descent, Mini-batch Gradient Descent, and Stochastic Gradient Descent. (6 pt)

A. **Batch Gradient Descent** uses the entire dataset to compute gradients in each step.

B. **Mini-batch Gradient Descent** processes data in small batches (e.g., size 32). In each step, you will compute the gradient and update the parameters based on just one mini-batch.

C. **Stochastic Gradient Descent (SGD)** updates the parameters after processing each single training example (i.e., a mini-batch of size 1).

ii. **Discussion:** Analyze the generated plot that visualizes the cost function over the number of updates for all three gradient descent variants. Discuss the trade-offs between the three methods. Compare them in terms of computational efficiency (speed) and stability of convergence. (1 pt)

Your solution here.