OSU CSE 5523
Homework #2: Problem Set

Release Date: Feb 1, 2024

Submission Instructions

Due Date: Feb 15 (23:59 ET), 2024

Submission: Please submit your solutions in a single PDF file named HW_2_name.number.pdf (e.g., HW_2_zhu.3440.pdf) to Carmen. You are **strongly** encouraged to use LaTeX to type your solutions. I include a Latex template (you don't need to type the problem), but you can use any other template. Otherwise, you may write your solutions on paper, or directly type your solutions and save them as a PDF file. *Submission in any other format will not be graded.* **For coding problems, please append your code to your submission and report your results (values, plots, etc.) in your written solution.** You will lose points if you only include them in your code submissions.

We highly recommend that you write down the derivation of your answers, and highlight your answers clearly!

Collaboration: Students are encouraged to work together on homework problems to develop an understanding of the material. However, each student **must**:

- generate and turn in his/her own **individual solutions** that reflect his/her own individual level of understanding (this includes computer programs), and
- **cite** any external resources used to answer a homework question (outside of the textbook, lecture notes, and other students in the course). Particularly, in your written report, you need to list with whom you have discussed for each problem (please do so in the first page).

In addition, students **may not**:

- **copy** homework solutions from another student or from any other source, or
- **consult** homework or exam solutions from previous offerings of this course.

Please consult the syllabus for what is and is not acceptable collaboration.

1 MLE [10 points]

Suppose $X_1, \ldots, X_n$ are i.i.d. samples from a uniform distribution on the interval $[-\theta, \theta]$, i.e., the pdf is given by

$$f(x) = \begin{cases} \frac{1}{2\theta}, & -\theta \leq x \leq \theta \\ 0, & \text{otherwise} \end{cases}$$

where the parameter $\theta > 0$ but is unknown. Please find maximum likelihood estimate (MLE) of θ.

	index i	1	2	3	4	5	6	7	8
Y	y_i	-1	-1	-1	+1	+ 1	+ 1	+ 1	+1
$X[1]$	$x_i[1]$	-1	-1	1	+ 1	+ 1	+1	-1	-1
$X[2]$	$x_i[2]$	1	2	3	1	1	1	2	3
$X[3]$	$x_i[3]$	1.0	2.0	3.0	1.0	1.0	1.0	2.0	3.0

2 Naive Bayes (simple practice) [15 points]

The table above shows a training set $D_{\text{tr}} = \{(\boldsymbol{x}_i, y_i \in \{+1, -1\})\}_{i=1}^{8}$ for binary classification, where $x_i[1] \in \{+1, -1\}$ (so binary), $x_i[2] \in \{1, 2, 3\}$ (so categorical), and $x_i[3] \in \mathbb{R}$. Let us fit a Naive Bayes model to it; i.e., we assume conditional independence in the sense that the value of a particular feature is independent of the value of any other feature, given the class variable. Therefore,

$$p(X = \boldsymbol{x}|Y = k) = \mathbb{P}(X[1] = x[1]|Y = k)\mathbb{P}(X[2] = x[2]|Y = k)p(X[3] = x[3]|Y = k), \quad \forall \quad k \in \{-1, +1\}.$$

We further assume $X[3]|Y = k$ is a Gaussian distribution with mean μ_k and variance σ_k^2 (i.e., μ_{+1}, σ_{+1}^2 for class $+1$ and μ_{-1}, σ_{-1}^2 for class -1). That is,

$$p(X[3] = x[3]|Y = k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x[3] - \mu_k)^2}{2\sigma_k^2}\right).$$

With this assumption, we will derive the *maximum likelihood estimates* for all the parameters (see lecture 7 for likelihood) and then apply the Bayes classifier.

1. [1 points] Write down the Bayes classifier (here also called Naive Bayes)

$$h^*(\boldsymbol{x}) = \underset{k \in \{-1,+1\}}{\arg\max} \_\_\_\_\_$$

2. [1 points] What is the value of $\mathbb{P}(Y = +1)$?
3. [1 points] What is the value of $\mathbb{P}(x[1] = -1|Y = +1)$?
4. [1 points] What is the value of $\mathbb{P}(x[2] = 1|Y = +1)$?
5. [1 points] What is the value of $\mathbb{P}(x[2] = 3|Y = -1)$?
6. [1 points] What is the value of $\mathbb{P}(x[2] = 3|Y = +1)$?
7. [1 points] What is the value of μ_{+1}?
8. [1 points] What is the value of σ_{+1}^2?
9. [1 points] What is the value of $p(x[3] = 1.5|Y = +1)$?
10. [1 points] What is the value of μ_{-1}?
11. What is the value of σ_{-1}^2?
12. [1 points] What is the value of $p(x[3] = 0.0|Y = -1)$?
13. [3 points] What is the predicted label $h^*(\boldsymbol{x})$ based on Bayes classifier for $\boldsymbol{x} = [-1, 3, 2.5]^\top$?

3 Naive Bayes as linear classifiers [Optional, 5 points]

This problem is optional. Let us fit a Naive Bayes model for a binary classification with ($\boldsymbol{X} \in \mathbb{R}^d, Y \in \{+1, -1\}$), i.e.,

$$\mathbb{P}(Y = y|X = \boldsymbol{x}) = \frac{\mathbb{P}(Y = y)\prod_{j=1}^{d} p(X[j] = x[j]|Y = y)}{p(X = \boldsymbol{x})},$$

where $p(X[j] = x[j]|Y = y)$ (or simply $p(x[j]|y)$) is a Gaussian distribution $\mathcal{N}(\mu_{j,y}, \sigma_j^2)$. That is, we assume that for each dimension d, the two one-dimensional Gaussian distributions (one for each class) share the same variance σ_j^2, but with different means $\mu_{j,+1}$ and $\mu_{j,-1}$. For $\mathbb{P}(Y = y)$, let $\mathbb{P}(Y = +1) = \lambda$.

Please show (1) $\mathbb{P}(Y = y|X = \boldsymbol{x})$ can be re-written as $\frac{1}{1+e^{-y(\boldsymbol{w}^\top \boldsymbol{x}+b)}}$ ($y \in \{+1, -1\}$) [3 points]; (2) what the corresponding $\boldsymbol{w}$ and b are [2 points]. Your answers should be based on $\mu_{j,y}$, σ_j, and λ. For $\boldsymbol{w} \in \mathbb{R}^d$, you may simply write the expression of $w[j]$ for a specific j. This question shows you that under certain assumptions, Naive Bayes will lead to a linear classifier. Please write down your derivation with no more than 15 lines for each answer.

Hint: You do not need to perform maximum likelihood estimation, but assume that $\mu_{d,y}$ and σ_d are given and fixed. You may also use the so-called "law-of-total-probability" for $p(X = \boldsymbol{x})$:

$$p(\boldsymbol{x}) = \mathbb{P}(Y = -1)\prod_{j=1}^{d} p(x[j]| - 1) + \mathbb{P}(Y = +1)\prod_{j=1}^{d} p(x[j]| + 1)$$

4 Newton's method and RSS loss [10 points]

Given a training set $\{(\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R})\}_{i=1}^n$, please learn a linear regressor $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$ (i.e., without the bias term) by minimizing the RSS loss (see Equation 1) *using the Newton's method*. That is, let $\boldsymbol{w}' \in \mathbb{R}^D$ be the current guess of the solution, the Newton's method *iteratively* updates $\boldsymbol{w}'$ by $\boldsymbol{w}' - \boldsymbol{H}^{-1}(\boldsymbol{w}')\nabla E(\boldsymbol{w}')$, where $\boldsymbol{H}$ is the Hessian matrix of $E(\boldsymbol{w})$ at $\boldsymbol{w}'$ and $\nabla E(\boldsymbol{w})$ is the gradient of $E(\boldsymbol{w})$ at $\boldsymbol{w}'$.

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} E(\boldsymbol{w}) = \min_{\boldsymbol{w}\in\mathbb{R}^d} \sum_i (\boldsymbol{w}^\top \boldsymbol{x}_i - y_i)^2 \tag{1}$$

Please show that, no matter what initialization $\boldsymbol{w}' \in \mathbb{R}^D$ you choose, the Newton's method will arrive at the closed-form solution (you may assume that the closed-form solution exists) after *just one update*. Please show your derivation step-by-step with no more than 15 lines.

Note:

$$f(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} = \sum_{i=1}^d \sum_{j=1}^d w_i a_{ij} w_j$$

$$\nabla f(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{w})}{\partial w_1} \\ \cdots \\ \frac{\partial f(\boldsymbol{w})}{\partial w_d} \end{bmatrix} = (\boldsymbol{A} + \boldsymbol{A}^\top)\boldsymbol{w}$$

$$\begin{aligned} H(\boldsymbol{w}) &= \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{w})}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 f(\boldsymbol{w})}{\partial w_1 \partial w_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{w})}{\partial w_d \partial w_1} & \cdots & \frac{\partial^2 f(\boldsymbol{w})}{\partial w_d \partial w_d} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} + a_{11} & \cdots & a_{11} + a_{1d} \\ \vdots & \ddots & \vdots \\ a_{d1} + a_{11} & \cdots & a_{dd} + a_{dd} \end{bmatrix} \\ &= \boldsymbol{A} + \boldsymbol{A}^\top \end{aligned}$$

Hints: You may use $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ to represent the entire data set's inputs, and $\boldsymbol{y} = [y_1, y_2, \cdots, y_n]^\top \in \mathbb{R}^n$ to represent the entire data set's output labels.

5 Logistic regression [25 points]

In this problem you will explore implementations of two different solvers for logistic regression. To get started, download the file `lr-gd.py` from the Carmen.

1. [5 points] We will begin by implementing logistic regression using standard gradient descent for performing the maximum likelihood estimation step. Review the slides from lecture 09 and lecture 10 and make sure you understand what `lr-gd.py` is doing. Test out the file by experimenting a bit with the various parameters—especially the "step size" or learning rate α—to obtain a good convergence rate. I would recommend trying a wide variety starting with $\alpha \approx 1$ and ranging to $\alpha \ll 1$. You should also feel free to play around with other stopping criteria than those provided. For this problem, all you need to do is to report the value of α you used, the number of iterations required for convergence and the final negative log-likelihood (which is the last element of `cost`) for your choices of α.

2. [15 points] Using the notes from lecture 9, adapt `lr-gd.py` to implement Newton's method for solving this problem. This algorithm requires computing both the gradient and the Hessian at each iteration. The Hessian in this case is given by

$$\frac{\nabla^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_{i=1}^{n} \widetilde{\boldsymbol{x}}_i \widetilde{\boldsymbol{x}}_i^\top g(\boldsymbol{\theta}^\top \widetilde{\boldsymbol{x}}_i)(1 - g(\boldsymbol{\theta}^\top \widetilde{\boldsymbol{x}}_i)),$$

where g is defined as in the notes. (You may verify this for yourself, but you do not need to.) [Take extra care in your implementation in making sure you have the correct signs in your algorithm—remember we are trying to *minimize* the *negative log-likelihood*. If your algorithm is not converging, this is a likely suspect. Note also that in the notes I assume that the $\widetilde{\boldsymbol{x}}$ are column vectors, but in the Python code they are treated as row vectors.]

Just as a reminder, in Python `A*B` computes the element-wise multiplication, while `A.dot(B)` computes matrix multiplication. You may also want to compute a matrix inverse of `A` using `np.linalg.inv(A)`. You should complete this assignment by writing a new function to replace `grad_desc`. Submit the code you produce for this function, and report the number of iterations required for convergence and the final negative log-likelihood (which is the last element of `cost`). Compare this to the results from part (a).

3. [5 points] Finally, try comparing the results from parts (a) and (b) when applied to a much larger dataset. Try changing `n_samples` to 100000. You will want to comment out the plotting portion of the code for this part (or at least the part that plots the training data). Record the final loss, running time and the number of iterations for both algorithms when applied to this dataset. Note that in order to run traditional gradient descent on this large of a dataset, you will need to make α much smaller (think about why this would be the case). If you see warnings regarding a divide by zero or the cost is much larger than the one obtained by Newton's method (or even `inf`), you have set α to be too large. Also, note that you can get the system time in Python by adding `import time` to your file and then using the `time.time()` command.