COMP6321: Machine Learning

Assignment 1

Due: 11:59 PM (EST), Sept 26, 2024, submit on Moodle.

Include your name and student ID!

- Submit your write-up in PDF (we do not accept hand-written submissions) and all source code as an ipynb file (with proper documentation). Your writeup should include written answers to all questions, including the reported results and plots of coding questions in a single PDF file. Please save the output of each cell or your coding questions may NOT be graded. Write a script for each programming exercise so the TAs can easily run and verify your results. Make sure your code runs!
- Text in square brackets are hints that can be ignored.

NOTE For all the exercises, you are only allowed to use the basic Python, Numpy, and matplotlib (or other libraries for plotting), unless specified otherwise.

Exercise 1: Perceptron (7 pts)

NOTE This is a theoretical question, and you are not required to code anything.

1. (1.5 points) Consider the dataset shown in Table 1 where $\mathrm{x} = (x_1, x_2) \in \{0,1\} \times \{0,1\}, y \in \{-,+\}$. Note that this dataset corresponds to the boolean function "AND" over the 2-bit binary input. Suppose we are training a Perceptron to learn on this dataset and we initialize the weights and the bias, $w^{(0)} = 0$ and $w_0^{(0)} = 0$. Please i) answer if this dataset is learnable by a Perceptron, and ii) if so, write down the weights update procedure for each iteration; if not, explain why.
2. (1.5 points) Extending AND to any boolean functions over a 2-bit binary input, where we have $2^{2^2} = 16$ possible distinct boolean functions in total, among which how many of them can be learnable by a Perceptron? Please also write down the truth table(s) of the boolean functions that are **not learnable**, if there are any. [Hint AND is one of the 16 boolean functions, with the output -, -, -, +. Similarly, a constant function represents two of the 16 boolean functions, with the output -, -, -, - or +, +, +, +.]
3. (4 points) *Modified Perceptron Algorithm* Recall the Perceptron algorithm we learned in class: for each "mistake" we update the weights by setting $w \leftarrow w + y_i \mathrm{x}_i, w_0 \leftarrow w_0 + y_i$. Now we would like to modify this algorithm by considering a constant $c > 1$ and making modifications to the update rule $w \leftarrow w + c y_i \mathrm{x}_i, w_0 \leftarrow w_0 + c y_i$. Let's call this algorithm "modified Perceptron". Please **prove or refute** that the modified Perceptron algorithm converges after the same number of mistakes as made by the Perceptron algorithm. [Hint: c.f. the convergence theorem in lecture 2]

x_1	x_2	y
0	0	-
0	1	-
1	0	-
1	1	+

Table 1: The Truth Table for AND Function.

Exercise 2: Logistic Regression (9 pts)

In this exercise you will implement the logistic regression algorithm and evaluate it on the dataset provided with this assignment. The training dataset is divided into five different csv files. You need to combine this into a single training dataset. Do not use any machine learning library, but feel free to use libraries for linear algebra and feel free to verify your results with existing machine learning libraries.

1. (2 pts) Let $\Pr(C_1|x) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$ and $\Pr(C_2|\mathbf{x}) = 1 - \sigma(\mathbf{w}^T\mathbf{x} + w_0)$. Learn the parameters $\mathbf{w}$ and w_0 using the gradient descent algorithm. Use the maximum number of epochs to be 100 for GD. You can also use any appropriate convergence criteria to stop the GD loop before 100 epochs. Use a step size of 0.1 (or 0.01 if it is converging poorly) for GD.

2. (0.25 pts) After the training process, create the following plots:
   (a) test error vs the number of epochs
   (b) training error vs the number of epochs
   (c) test loss vs the number of epochs
   (d) training loss vs the number of epochs
   (e) Print the parameters $\mathbf{w}, w_0$ found for logistic regression.
3. (5 pts) Now let's add a regularization term $0.5\lambda||\mathbf{w}||_\mathbf{2}^\mathbf{2}$ to the loss function of your logistic regression algorithm. Your new loss function will be $\mathcal{L}_{new} = \mathcal{L}_{old} + 0.5\lambda||\mathbf{w}||_\mathbf{2}^\mathbf{2}$, where $\mathcal{L}_{old}$ is the loss function of logistic regression we learned in lecture 3. Choose two values of $\lambda = \{0.5, 1\}$ and train this algorithm on the given dataset with these two values of λ.
4. (0.25 pts) After the training process, create the plots mentioned in step 2, for this updated logistic regression algorithm. You should have two sets of plots since you used two different values of *lambda* to train the model.
5. (1.5 pts) Compared the results for step 2 with the results for the updated logistic regression with two values of *lambda*. Which of the tree algorithms performs the best? Why? Compare the values of $\mathbf{w}, w_0$ for all three cases. How is λ affecting these parameter values, and the overall error?
6. (Extra Credit: 1.5 pts) Use 5-fold cross-validation on the training dataset to find the best value for λ. Report the best λ, and also draw a plot that shows the cross-validation error of logistic regression as λ varies. Note that the training dataset is already divided into 5 different csv files, to ease the cross-validation process. Report the test error for the best λ. Is this better than the results in steps 2, and 5? Why or why not?