Due: 05/22/2024 23:59 on NTU COOL

Foundations of Artificial Intelligence: Homework 3

Instructor: Shang-Tse Chen & Yun-Nung (Vivian) Chen

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for misconduct. Discussions on course materials and homework solutions are encouraged, but you should write the final solutions alone and understand them fully. Books, notes, Internet, ChatGPT resources can be consulted but not copied from. Please list your references to avoid any plagiarism concerns.

1 Hand-written Part

Problem 1 (10 points)

Consider a binary classification data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{-1, +1\}$. For any weight vector $\mathbf{w}$ within a linear model, define an error function

$$\text{err}(\mathbf{w}^T\mathbf{x}, y) = \big(\max(1 - y\mathbf{w}^T\mathbf{x}, 0)\big)^2.$$

That is,

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\big(\max(1 - y_n\mathbf{w}^T\mathbf{x}_n, 0)\big)^2$$

Running gradient descent to optimize $E_{\text{in}}(\mathbf{w})$ requires calculating its gradient direction $\nabla E_{\text{in}}(\mathbf{w})$ (and then move opposite to that direction). What is $\nabla E_{\text{in}}(\mathbf{w})$?

The error function err *is called the squared hinge error and is a core component in the so-called l_2-loss SVM.*

Problem 2 (15 points)

Consider a process that generates d-dimensional vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ independently from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{u}, \mathrm{I})$, where $\mathbf{u} \in \mathbb{R}^d$ is an unknown parameter vector and $\mathrm{I} \in \mathbb{R}^{d\times d}$ is an identity matrix. The maximum likelihood estimate of $\mathbf{u}$ is

$$\mathbf{u}^* = \arg\max_{\mathbf{u}\in\mathbb{R}^d} \prod_{n=1}^{N} p_{\mathbf{u}}(\mathbf{x}_n),$$

where $p_{\mathbf{u}}$ is the probability density function of $\mathcal{N}(\mathbf{u}, \mathrm{I})$. Prove that

$$\mathbf{u}^* = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n.$$

The same derivation can be used to obtain the cross-entropy error function of logistic regression.

Problem 3 (15 points)

A classic binary classification data set that cannot be separated by any line is called the XOR data set, with

$\mathbf{x} = [x_1, x_2]$	y
$\mathbf{x}_1 = [+1, +1]$	$y_1 = -1$
$\mathbf{x}_2 = [-1, +1]$	$y_2 = +1$
$\mathbf{x}_3 = [-1, -1]$	$y_3 = -1$
$\mathbf{x}_4 = [+1, -1]$	$y_4 = +1$

You can see why it is called XOR by interpreting $+1$ as a boolean value of TRUE and -1 as FALSE. Consider a second-order feature transform $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ that converts the data set to

$\mathbf{z} = \Phi_2(\mathbf{x})$	y
$\mathbf{z}_1 = \Phi_2(\mathbf{x}_1)$	$y_1 = -1$
$\mathbf{z}_2 = \Phi_2(\mathbf{x}_2)$	$y_2 = +1$
$\mathbf{z}_3 = \Phi_2(\mathbf{x}_3)$	$y_3 = -1$
$\mathbf{z}_4 = \Phi_2(\mathbf{x}_4)$	$y_4 = +1$

Show a perceptron $\tilde{\mathbf{w}}$ in the $\mathcal{Z}$-space that separates the data. That is,

$$y_n = \text{sign}(\tilde{\mathbf{w}}^T \mathbf{z}_n) \text{ for } n = 1, 2, 3, 4.$$

Then, plot the classification boundary

$$\tilde{\mathbf{w}}^T \Phi_2(\mathbf{x}) = 0$$

in the $\mathcal{X}$-space. Your boundary should look like a quadratic curve that classifies $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ perfectly.

Problem 4 (15 points)

Consider building binary classification with AdaBoost algorithm, a weak classifier $g_t(\mathbf{x})$ is trained on a data set $\{\mathbf{x_n}, y_n\}_{n=1}^N$ with weights $\{w_n^t\}_{n=1}^N$ at the time step t. The error rate is defined as

$$\epsilon_t = \frac{\sum_{n=1}^N w_n^t \cdot \delta\big(g_t(\mathbf{x_n}), y_n\big)}{\sum_{n=1}^N w_n^t},$$

where $\delta\big(g_t(\mathbf{x_n}), y_n\big) = 1$ if $g_t(\mathbf{x_n}) \neq y_n$ and $\delta\big(g_t(\mathbf{x_n}), y_n\big) = 0$ otherwise. For the next time step, the data set is reweighed to emphasize on misclassified samples through the following rules

$$w_n^{t+1} = \begin{cases} w_n^t \cdot d_t & \text{if } g_t(\mathbf{x_n}) \neq y_n \\ w_n^t / d_t & \text{if } g_t(\mathbf{x_n}) = y_n \end{cases}.$$

Show that $d_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$ can degrade the previous classifier $g_t(\mathbf{x})$ and make its error rate become 0.5 with new weights $\{w_n^{t+1}\}_{n=1}^N$:

$$\frac{\sum_{n=1}^N w_n^{t+1} \delta\big(g_t(\mathbf{x_n}), y_n\big)}{\sum_{n=1}^N w_n^{t+1}} = 0.5.$$

2 Programming Part

The goal of this programming part is to implement a supervised machine learning pipeline from scratch, which includes preprocessing, training, and evaluation of linear and nonlinear models on a given dataset. You will apply this pipeline to a classification task and a regression task.

You **MUST** use the provided sample code `hw3.py` from NTU Cool or here as a starting point for your implementation. For the core algorithm implementation, you are only allowed to use `pandas`, `numpy`, and Python Built-in libraries. There are no library restrictions for data analysis or plotting.

2.1 Homework Description

Complete the following tasks:

1. **Dataset**
   (a) Classification task: Iris dataset (https://archive.ics.uci.edu/ml/datasets/iris)
   (b) Regression task: Boston Housing dataset (https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv)
2. **Preprocessing**
   (a) Load and split the dataset into train and test sets (70% train, 30% test).
   (b) Implement feature scaling: normalize or standardize features based on your choice.
3. **Models**
   (a) Implement a linear model (logistic regression for classification, linear regression for regression).
   (b) Implement a nonlinear model (decision tree for classification, decision tree for regression).
   (c) Implement a random forest model for classification and regression).
4. **Training**
   (a) Implement gradient descent for logistic and linear regression.
   (b) Implement a basic algorithm for building a decision tree.
   (c) Implement a basic algorithm for building a random forest.
5. **Evaluation**
   (a) Implement evaluation metrics: accuracy for classification, mean squared error (MSE) for regression.
   (b) Compare the performance of linear and nonlinear models on both datasets.
6. **Report**

2.2 Deliverables

1. Source code (Python)
2. A detailed report answering the following questions:
   (a) Compare the performance of the linear, nonlinear, and random forest models on the classification and regression tasks. Based on the evaluation metrics (accuracy for classification and mean squared error for regression), which model performs better for each task, and why do you think this is the case?
   (b) Apply both normalization and standardization techniques to the datasets and compare their impact on the performance of one of the models (e.g., logistic regression). Explain the rationale behind each technique, and discuss their advantages and disadvantages in the context of the given tasks.
   (c) Train your logistic or linear regression model using two different configurations of learning rate and number of iterations. Compare the performance of these configurations using the evaluation metrics (accuracy for classification and mean squared error for regression). Discuss how the choice of learning rate and the number of iterations affect the convergence and overall performance of the models.
   (d) Discuss the impact of hyperparameters, such as the number of trees and maximum depth, on the performance of the random forest model for both classification and regression tasks. How do these hyperparameters affect the model's complexity, generalization, and potential for overfitting? Include a brief explanation of how you selected or tuned these hyperparameters in your implementation.
   (e) Analyze the strengths and weaknesses of the implemented models. In what scenarios would you choose to use a linear model over a nonlinear model, or a random forest model, and vice versa? Discuss the trade-offs between model complexity, interpretability, and performance.

2.3 Grading Criteria

Criterion 1 Preprocessing implementation: 2%

Criterion 2 Linear model implementation: 12%

Criterion 3 Nonlinear model implementation: 14%

Criterion 4 Random forest implementation: 5%

Criterion 5 Evaluation metric implementation: 2%

Criterion 6 Report quality and clarity: 10%

Submission

Write a PDF report for P1-P4 and Programming Part. Complete hw3.py for Programming Part.

Then, submit a zip file to NTU COOL. The zip file should be named `b0x902xxx.zip` that contains a directory called `b0x902xxx/`, which includes two files, the completed `hw3.py` and `report.pdf`.