CS 559: Machine Learning: Fundamentals and Applications

Assignment 2 Due: 10/1/2024 Tuesday 11:59 p.m.

- The assignment must be individual work and must not be copied or shared. Any tendency to cheat/copy evidence will lead to a 0 mark for the assignment.
- Students must only use Pandas, NumPy, Matplotlib, and Scipy if the problem does not specify libraries/packages.
- All codes will be tested in grading. Any codes with an error will be marked 0. Make sure to restart the kernel and run it all before the submission. Delete any codes that do not want to be graded.
- Results must be displayed.
- All problems must be submitted in a single notebook file. Do not use a text editor to write codes.

1 Naive Bayes Classification [40 pts]

Use the following code to generate the train data set. The code will generate a random data set with four features and classes.

```
from sklearn import datasets
X, y = datasets.make_blobs(n_samples = 400, n_features= 5,
    centers = 4, cluster_std= 2, random_state= 100)
```

a. [5 pts] Compute the prior probability of each class, $p(C_k)\ \forall k = 1, \ldots, 4$.
b. [10 pts] Compute the likelihood $p(\mathbf{X}|C_k)\ \forall k = 1, \ldots, 4$.
c. [15 pts] Compute the posterior probability of each point $p(C_k|\mathbf{X})\ \forall k = 1, \ldots, 4$. Assign the class ID to each point.
d. [5 pts] Construct the confusion matrix to show the classification rate using *sklearn.metrics.confusion_matrix*. The confusion matrix visualizes and summarizes the performance of a classification algorithm, as shown below.

Total Prediction = 400	$y = 0$	$y = 1$	$y = 2$	$y = 3$
$\hat{h} = 0$				
$\hat{h} = 1$				
$\hat{h} = 2$				
$\hat{h} = 3$				

e. [5 pts] Classify the target using **sklearn.native_bayes.GaussianNB**. Report the accuracy of the model.

2 Perceptron [30 pts]

In the lecture, the implementation of the perception algorithm was discussed with pseudo-code. In this problem, students will implement the algorithm and generalize the model using the same data set used in problem 1.

a. [15 pts] In the lecture slide, the methods needed for the perceptron algorithm are provided (**step(X)** and **perceptron_predict(w, X)**). Write a method (**Perceptron_fit(w, X, y, learning_rate, iteration)**) that fits the data and returns **w**.
b. [10 pts] Create a sample of **X** in Question 1 whose $y \in \{0, 1\}$. Fit the sample data and find **w** when the learning rate is 0.001 and iteration is 1. The learning rate and iteration numbers can be tuned to increase the performance if necessary.
c. [5 pts] Using the final **w** from 2.b, to classify all observations in **X**. Measure the performance and explain the success. Discuss what other tests can be done in order to improve the classification for **X**.

3 Logistic Regression Regularization [30 pts]

One of the ways to avoid overfitting is by **regularizing** the weights. In this problem, we will observe how the model gets regularized using the same data set from Question 1.

a. [10 pts] Regularize a *sklearn.linear_model.LogisticRegression* model with a hyperparameter **penalty**='l1' (Lasso regularization). In the regularization validation, there is a hyperparameter to be concerned about. The model needs to be trained iteratively by tunning a hyperparameter, **C**= λ^{-1}, whose default value is 1. A hyperparameter **solver='saga'**, which is one of the fastest convergence solvers for any regularization in LogisticRegression(), must be used when 'l1' penalty parameter is used.

```
from sklearn.linear_model import LogisiticRegression
clf = LogisiticRegression(penalty = 'l1',
    solver = 'saga', C = 1.0)
...
w, w0 = clf.coef_, clf.intercept_
```

Make a plot of **w** vs. C to show the convergence of **w** as C changes.

b. [10 pts] Explain which feature is the most important in each class.

c. [10 pts] Do a similar as done in a) to show that Ridge regularization (penalty='l2') does not provide a sparse solution.