CS 559: Machine Learning: Fundamentals and Applications

Due: 3/7/2024 Thursday 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, and Spacy if the coding problem does not specify libraries/packages. Use of other libraries than specified will be penalized.
- All problems must be submitted in a single notebook file.

1 Logistic Regression [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 **HW3**.

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

a. [10 pts] Regularize a Sklearn.linear_model.LogisticRegression model with a hyperparameter **penalty**='l1' (Lasso regularization) to visualize the convergence of weights as a regularization term, λ, changes. 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_
```

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') always has a sparse solution. Explain if a sparse solution is expected.

2 Neural Network Regression [40 pts]

Consider neural networks with one hidden layer, as discussed during the lecture. The forward propagation prediction was implemented. Complete the backward propagation codes. Using the given example in the lecture.

```
x = np.array([0.7, 0.1, 0.3, 0.5]).reshape(1,4)
y = np.array([1.5]).reshape(-1,1)
w1 = np.array([[0.16, 0.16],[0.02, 0.25],[0.63, 0.22],[0.36, 0.29]])
w2 = np.array([0.05, 0.33])
```

a. (10 pts) Perform the backward propagation from the output layer to the hidden layer. Compute δ_2 and $\frac{dE}{dW_2}$. Then, update W_2. Use the learning rate $\eta = 0.1$.

b. (15 pts) Perform the backward Propagation from the hidden layer to the input layer. Compute δ_1 and $\frac{dE}{dW_1}$. Then, update W_1. Use the learning rate $\eta = 0.1$.

c. (10 pts) Put the forward and backward propagation codes together in a for-loop to observe the error convergence.

d. (5 pts) Use Scikit-learn Neural Network to predict the target variable. Any hyperparameters can be tuned as shown in the notebook file.