ECE 473 : Introduction to Artificial Intelligence

Assignment 3

Instructions on programming assignments

You are asked to modify the code in `hw3_submission.py` between

```
##################################################################
##################################################################
# BEGIN_YOUR_CODE
# HINTS OR INSTRUCTIONS

pass;

# END_YOUR_CODE
##################################################################
##################################################################
```

- Please use Python3 to do all assignments in the course.
- You are allowed to make your own helper functions outside the skeleton functions.
- Do not change other than `hw3_submission.py`. You are only allowed to use `numpy` package to compute and construct functions. Do not use `scikit-learn, pytorch`, or `tensorflow` etc to implement your functions.
- You can test your code with test functions in `test.py`. For your reference, you can compare your own classifiers with the ones in sklearn package that is implemented in `test.py`.
- Homework will be graded based on the provided dataset and may be additionally tested on datasets that are not released to students.
- You need to get no more than 5% difference of accuracy from the reference to get full score.
- Only submit `hw3_submission.py`

Problem 1: Gaussian Naive Bayes Classifier

Calculation of Bayes Theorem can be simplified by making some assumptions, such as independence between all input variables. Although a dramatic and unrealistic assumption, this has the effect of making the calculations of the conditional probability tractable and results in an effective classification model referred to as Naive Bayes.

The Naive Bayes algorithm has proven effective and therefore is popular for text classification tasks. The words in a document may be encoded as binary (word present), or count (word occurrence) input vectors and binary, multinomial, or Gaussian probability distributions used.

In this problem, you will implement Gaussian Naive Bayes Classifier and test it on two datasets:

1. breast cancer dataset
2. digits dataset

Digits dataset consist of 1797 samples with 10 classes. Each sample has 64 features, which is originally 8×8 image. Your task for both dataset is to predict correct class for the test samples.
(Note: When you compute Bayes rule, denominator cannot be 0. In other words, a value inside `log` cannot be equal or less than 0.)

Tasks in Problem 1:

1. Implement function `Naive_Bayes.mean`
2. Implement function `Naive_Bayes.std`
3. Implement function `Naive_Bayes.gen_by_class`
4. Implement function `Naive_Bayes.calc_gaussian_dist`
5. Implement function `Naive_Bayes.predict`

Problem 2: Two-Layered Neural Network

Based on the structure of Logistic regression of Homework 2, you will extend the structure to two-layers neural network in this problem. Note that the counting index of "layers" starts with the first hidden layer up to the output layer. Thus, as illustrated in Fig 1, two-layered neural network has one hidden layer with a certain number of hidden units. The number of hidden units can vary considering various aspects such as how complex the input feature is or how deep the neural network is.

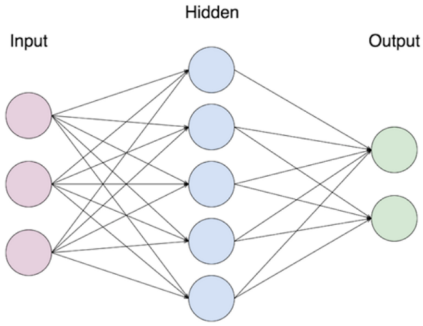

Figure 1: Illustration of two-layered neural network.

In this homework, you will implement a two-layered neural network with a hidden layer with 64 units and ReLU activation before fed forward to the output layer. Whereas for the output layer, you will use `sigmoid` function instead.

Denote $\mathbf{x} \in \mathbb{R}^D$ as an input data with D dimensions and $y \in \{0, 1\}$ as the corresponding true binary label. We denote $f(\mathbf{x}; \theta) : \mathbb{R}^D \to \mathbb{R}$ as a two-layered neural network that we want to train. The goal is to find f that minimizes the loss $\mathcal{L}$ in Eq.(1).

$$\mathcal{L}(f(\mathbf{x}), y) = \mathcal{L}_{BCE}(f(\mathbf{x}), y) + \lambda \|\theta\|_2^2 \,. \tag{1}$$

This loss $\mathcal{L}$ consist of two terms: 1) binary cross entropy loss ($\mathcal{L}_{BCE}$) and 2) regularization. The first term $\mathcal{L}_{BCE}$ is identical to the logistic regression in hw2 as in Eq.(2). The second term is the regularization with the weight λ which is for better generalization and to prevent the model from over-fitting to the training data. **Note that θ includes all trainable parameters including weights and biases in all layers.** To update the parameters with back propagation in the neural network, you will use mini-batch stochastic gradient descent method as in Homework 2.

$$\mathcal{L}_{BCE}(f(\mathbf{x}), y) = \mathbb{E}_{(\mathbf{x},y)}\Big[y \log(f(\mathbf{x})) + (1 - y) \log(1 - f(\mathbf{x}))\Big] \,. \tag{2}$$

Note: hyperparameters are provided as follows. Do not change the hyperparameter setting.

- learning rate = 10^{-3},
- batch size = 64,
- iterations number = 2000,
- regularization hyperparameter λ in Eq. (1) (i.e., `reg` in `hw3_submission.py`) = 10^{-3}.

In this problem, you will test the neural network on breast cancer dataset.

Tasks in Problem 2:

1. Implement function `Neural_Network.activation`
2. Implement function `Neural_Network.sigmoid`
3. Implement function `Neural_Network.loss`

4. Implement function `Neural_Network.fit`

5. Implement function `Neural_Network.predict`