Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2

з дисципліни «Технології машинного навчання»

Тема:

«Функції помилок (втрат) у машинному навчанні»

Виконав студент групи ІА-12:
Бутрій Віталій

Перевірив:
Коломоєць С.О.

Київ 2024

Мета роботи: отримати знання і критерії застосування основних використовуваних у сучасному машинному навчанні функцій помилок (функцій втрат)

Завдання

1) Завантажте дані за посиланням https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Для цього створив окремий клас, який буде завантажувати дані та розділяти дані на навчальну та тестову вибірки. Для цього використаю знайомі нам бібліотеки sklearn та pytorch:

```
class DataHandler:
    def __init__(self, file_path):
        self.file_path = file_path

    def load_data(self):
        column_names = ['Variance', 'Skewness', 'Curtosis', 'Entropy',
'Class']
        data = pd.read_csv(self.file_path, header=None, names=column_names)
        X = data.drop('Class', axis=1)
        y = data['Class']
        return X, y

    @staticmethod
    def split_data(X, y, test_size=0.2):
        return train_test_split(X, y, test_size=test_size, random_state=42)

    @staticmethod
    def prepare_data(X_train, y_train, X_test, y_test):
        X_train_tensor = torch.tensor(X_train.values, dtype=torch.float32)
        y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32)
        X_test_tensor = torch.tensor(X_test.values, dtype=torch.float32)
        y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32)

        train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
        test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
```

```
        train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
        test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

2) Реалізувати модель логістичної регресії з наступними функціями втрат:

а) Logistic loss

б) Adaboost loss

в) binary crossentropy

Для цього реалізую клас з функціями витрат та клас навчання моделей (які будуть, відповідно, використовувати ці функції):

```
class LossFunctions:
    @staticmethod
    def logistic_loss(output, target):
        output = torch.clamp(output, min=1e-7, max=1 - 1e-7)
        return - (target * torch.log(output) + (1 - target) * torch.log(1 -
output)).mean()

    @staticmethod
    def adaboost_loss(output, target):
        target = 2 * target - 1
        return torch.exp(-target * output).mean()

    @staticmethod
    def binary_crossentropy_loss(output, target):
        criterion = nn.BCELoss()
        return criterion(output, target)

class Trainer:
    def __init__(self, model, criterion, optimizer, train_loader, test_loader,
epochs=20):
        self.model = model
        self.criterion = criterion
        self.optimizer = optimizer
        self.train_loader = train_loader
        self.test_loader = test_loader
        self.epochs = epochs
```

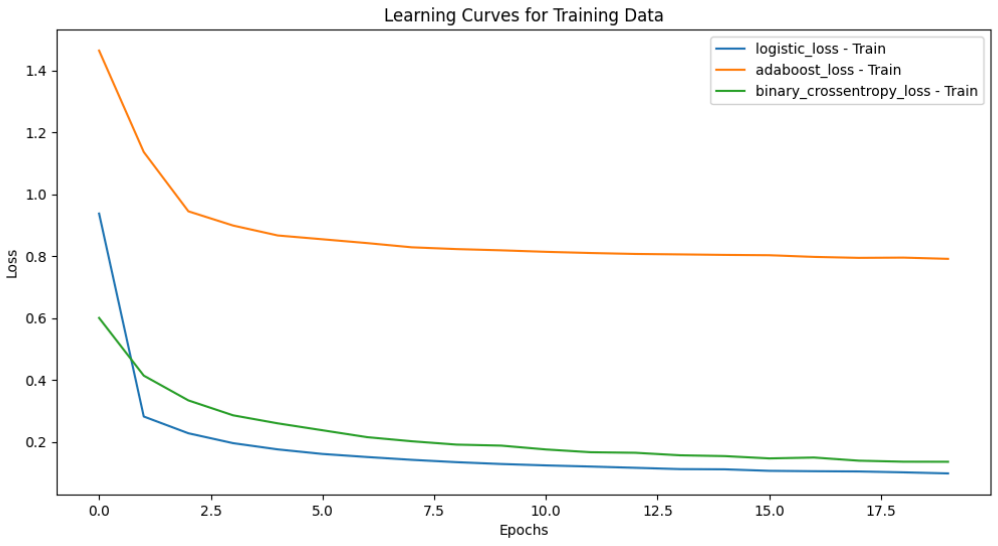
```
    def train(self):
        train_loss_history = []
        test_loss_history = []

        for epoch in range(self.epochs):
            self.model.train()
            epoch_train_loss = 0

            for inputs, labels in self.train_loader:
                self.optimizer.zero_grad()
                outputs = self.model(inputs)
                loss = self.criterion(outputs.squeeze(), labels.float())
                loss.backward()
                self.optimizer.step()
                epoch_train_loss += loss.item()

            train_loss_history.append(epoch_train_loss /
len(self.train_loader))

            self.model.eval()
            epoch_test_loss = 0

            with torch.no_grad():
                for inputs, labels in self.test_loader:
                    outputs = self.model(inputs)
                    loss = self.criterion(outputs.squeeze(), labels.float())
                    epoch_test_loss += loss.item()

            test_loss_history.append(epoch_test_loss / len(self.test_loader))

        return train_loss_history, test_loss_history

    def evaluate(self):
        self.model.eval()
        predictions, targets = [], []

        with torch.no_grad():
            for inputs, labels in self.test_loader:
                outputs = self.model(inputs)
                predicted = outputs.squeeze().round()
```

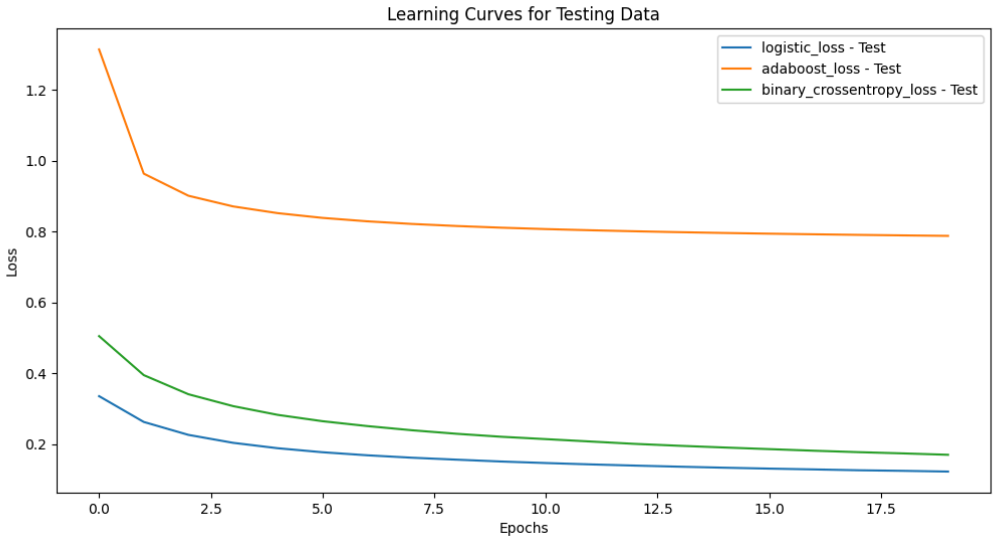
```
            predictions.extend(predicted.numpy())
            targets.extend(labels.numpy())

    return accuracy_score(targets, predictions)
```

3) Візуалізувати криві навчання моделі бінарної класифікації у вигляді динаміки зміни кожної з функцій помилок п.2 на тренувальній та тестовій вибірках.

Для порівняння функцій втрат і точності ми будуємо графіки навчальних і тестових втрат для кожної з функцій втрат.

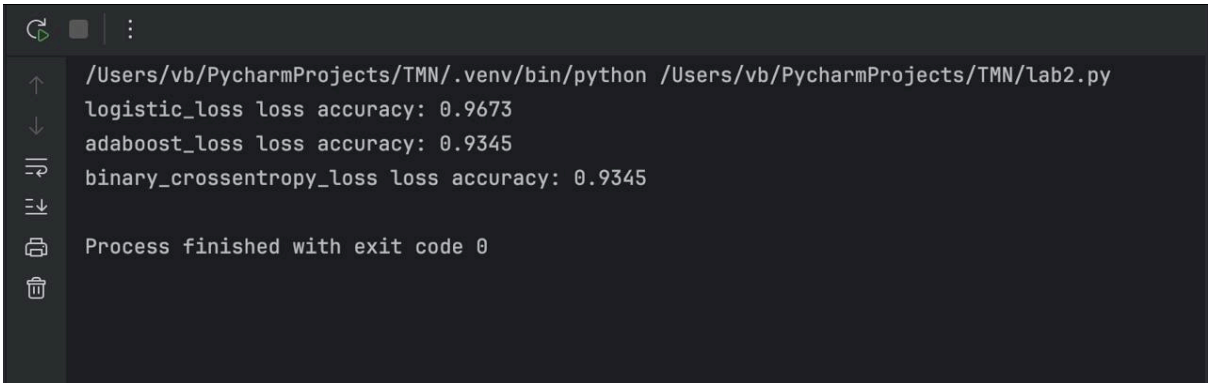
Це дозволяє наочно побачити динаміку зміни помилок і порівняти ефективність різних варіантів функцій втрат на тренувальних і тестових даних.





4) Порівняти якість класифікації за метрикою accuracy у кожному з трьох модифікацій алгоритму

Маємо такий аутпут:



Висновок:

У цій лабораторній роботі було реалізовано та проаналізовано основні функції втрат, зокрема логістичну функцію та бінарну крос-ентропію, які часто застосовуються в задачах бінарної класифікації. Було показано

відмінності в їхньому використанні та вплив на процес навчання моделі, що дало змогу краще зрозуміти критерії вибору функції втрат для конкретних задач машинного навчання.