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

Лабораторна робота №2

Технології машинного навчання

Тема: «Функції помилок (втрат) у машинному навчанні »

Виконав:
Студент групи ІА-12
Оверчук Дмитро Максимович

Перевірив:
Коломоєць С.О.

Київ 2024

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

Завдання на лабораторну роботу

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

2) Реалізувати модель логістичної регресії з наступними функціями втрат:

а) Logistic loss

б) Adaboost loss

в) binary crossentropy

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

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

Хід роботи

1. Підготовка даних

Дані з набору Banknote Authentication були завантажені та підготовлені за допомогою функцій:

```
def load_dataset():
    """
    Завантаження датасету та розділення на ознаки і мітки.
    """
    column_names = ['Variance', 'Skewness', 'Curtosis', 'Entropy', 'Class']
    data = pd.read_csv('data_banknote_authentication.txt', header=None,
names=column_names)
    X = data.drop('Class', axis=1)  # Ознаки
    y = data['Class']  # Цільова змінна
    return X, y
```

2. Модель логістичної регресії

Модель була реалізована у файлі logistic_regression_model.py:

```
class LogisticRegressionModel(nn.Module):
    def __init__(self, input_size):
        super(LogisticRegressionModel, self).__init__()
        self.linear = nn.Linear(input_size, 1)  # Один лінійний шар для
прогнозування

    def forward(self, x):
        return torch.sigmoid(self.linear(x))  # Застосування сигмоїдної функції
для отримання ймовірностей
```

3. Функції втрат

Функції втрат визначені як:

- Logistic Loss:

```
# Обчислення логістичних втрат (logistic loss)
def compute_logistic_loss(output, target):
    output = torch.clamp(output, min=1e-9, max=1 - 1e-9)  # Уникнення log(0)
через обмеження значень
    loss = - (target * torch.log(output) + (1 - target) * torch.log(1 -
output))  # Формула логістичних втрат
    return loss.mean()  # Середнє значення втрат
```

- AdaBoost Loss:

```
# Функція для обчислення втрат AdaBoost
def compute_adaboost_loss(output, target):
    target = 2 * target - 1  # Конвертація міток у {-1, 1} для AdaBoost
    loss = torch.exp(-target * output)  # Обчислення втрат згідно з методом
AdaBoost
    return loss.mean()
```

- Binary Crossentropy:

```
# Обчислення втрат за допомогою вбудованої PyTorch функції BCE
def compute_binary_crossentropy_loss(output, target):
    criterion = nn.BCELoss()  # BCE функція втрат
    return criterion(output, target)
```

4. Навчання моделей

Навчання моделей було реалізоване у файлі model_training.py:

```
def train_model(model, loss_fn, optimizer, train_loader, test_loader,
num_epochs=20):
    """
    Тренування моделі з фіксацією втрат для кожної епохи.
    """
    train_losses, test_losses = [], []

    for epoch in range(num_epochs):
        model.train()
        total_train_loss = 0

        for inputs, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = loss_fn(outputs.squeeze(), labels.float())
            loss.backward()
            optimizer.step()
            total_train_loss += loss.item()
```

```
        train_losses.append(total_train_loss / len(train_loader))

        # Оцінка втрат на тестових даних
        model.eval()
        total_test_loss = 0
        with torch.no_grad():
            for inputs, labels in test_loader:
                outputs = model(inputs)
                loss = loss_fn(outputs.squeeze(), labels.float())
                total_test_loss += loss.item()

        test_losses.append(total_test_loss / len(test_loader))

    return train_losses, test_losses
```

5. Побудова графіків

Графіки кривих втрат будувалися для кожної функції:

```
# Побудова графіків втрат для тренування і тестування
def plot_loss_curves(loss_histories):
    """
    Візуалізація динаміки втрат для тренувальних і тестових даних.
    """
    plt.figure(figsize=(12, 6))
    for name, (train_loss, test_loss) in loss_histories.items():
        plt.plot(train_loss, label=f'{name} - Тренувальні втрати')

    plt.xlabel('Епохи')
    plt.ylabel('Втрати')
    plt.title('Криві тренувальних втрат')
    plt.legend()
    plt.show()

    plt.figure(figsize=(12, 6))
    for name, (train_loss, test_loss) in loss_histories.items():
        plt.plot(test_loss, label=f'{name} - Тестові втрати')

    plt.xlabel('Епохи')
    plt.ylabel('Втрати')
    plt.title('Криві тестових втрат')
    plt.legend()
    plt.show()
```

6. Оцінка точності

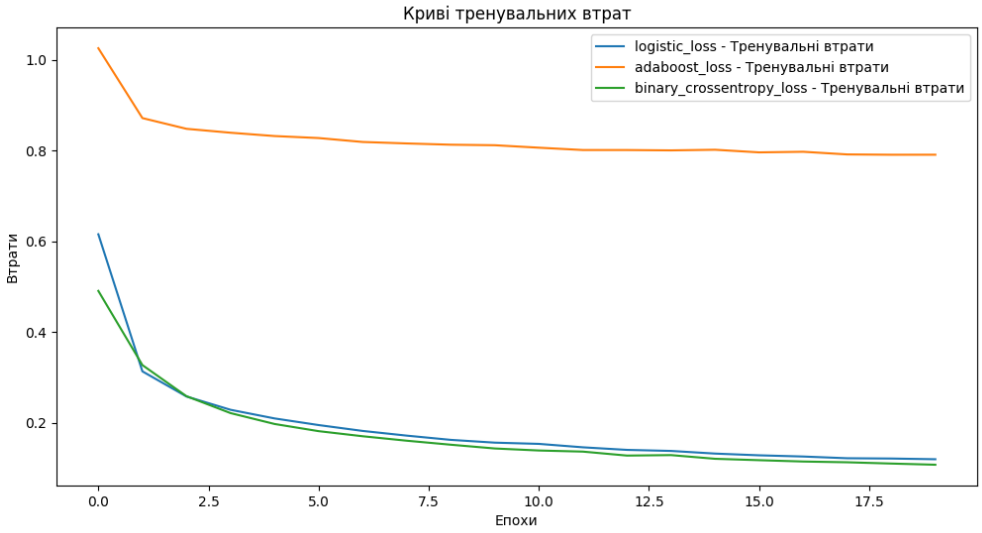
Точність моделей обчислювалася за метрикою accuracy:

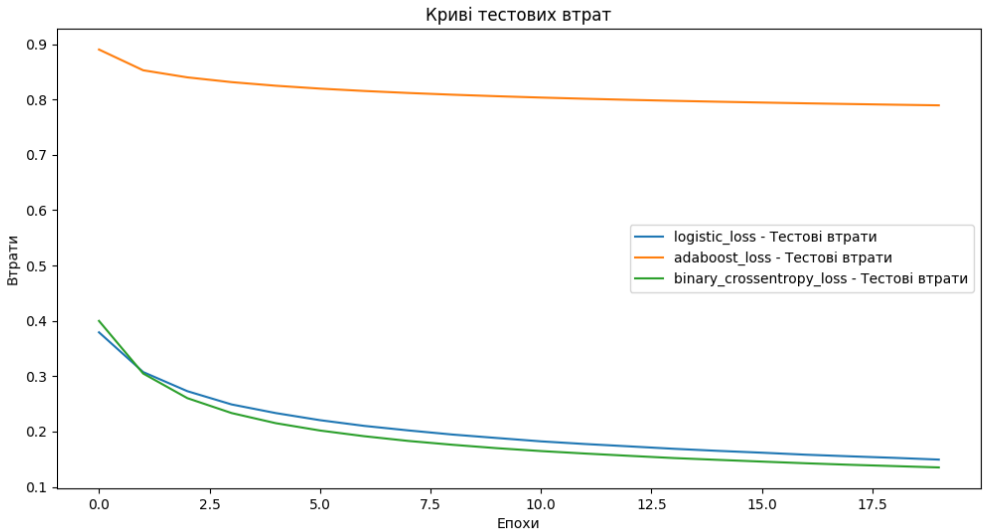
```
# Обчислення точності моделі
def calculate_model_accuracy(model, test_loader):
    """
    Оцінка точності моделі за тестовим набором даних.
    """
    model.eval()  # Встановлення моделі в режим оцінки
    predictions = []  # Список для прогнозів
    targets = []  # Список для реальних міток

    with torch.no_grad():  # Вимкнення обчислення градієнтів
        for inputs, labels in test_loader:
            outputs = model(inputs)  # Передбачення
            predicted = outputs.squeeze().round()  # Перетворення ймовірностей у
```

```
мітки {0, 1}
            predictions.extend(predicted.numpy())  # Збереження передбачень
            targets.extend(labels.numpy())  # Збереження міток

    return accuracy_score(targets, predictions)  # Повертає точність моделі
```

Результати:

```
Тренування моделі logistic_loss...
Точність моделі logistic_loss: 0.9455
Тренування моделі adaboost_loss...
Точність моделі adaboost_loss: 0.9018
Тренування моделі binary_crossentropy_loss...
Точність моделі binary_crossentropy_loss: 0.9636
```





Висновки: Binary Crossentropy показала найкращий результат за метрикою accuracy. AdaBoost Loss демонструє жорсткіший штраф для помилок, що вплинуло на точність. Використання різних функцій втрат дозволяє ефективно вирішувати задачі класифікації з урахуванням особливостей даних.