

Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Отчёт по лабораторной работе №4

Выполнил:
Студент 3 курса
Группы АС-65
Романюк Д. А.
Проверил:
Крощенко А. А.

Брест 2025

Цель: построить, обучить и оценить многослойный перцептрон (MLP) для решения задачи классификации.

Вариант 5

```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, LabelEncoder,
StandardScaler
from sklearn.metrics import accuracy_score

df = pd.read_csv(r"C:\OMO\3lab\mushrooms.csv")

y = df["class"]
X = df.drop("class", axis=1)

encoder = OneHotEncoder(sparse_output=False)
X_encoded = encoder.fit_transform(X)

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y_encoded, test_size=0.3, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train_t = torch.tensor(X_train, dtype=torch.float32)
y_train_t = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
X_test_t = torch.tensor(X_test, dtype=torch.float32)
y_test_t = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

class MLP(nn.Module):
    def __init__(self, input_size, activation_fn):
        super(MLP, self).__init__()
        self.hidden = nn.Linear(input_size, 8)
        self.activation = activation_fn
        self.output = nn.Linear(8, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.activation(self.hidden(x))
        x = self.sigmoid(self.output(x))
        return x

def train_and_evaluate(activation_fn, name):
    input_size = X_train_t.shape[1]
    model = MLP(input_size, activation_fn)
```

```

criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

epochs = 20
for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()
    y_pred = model(X_train_t)
    loss = criterion(y_pred, y_train_t)
    loss.backward()
    optimizer.step()

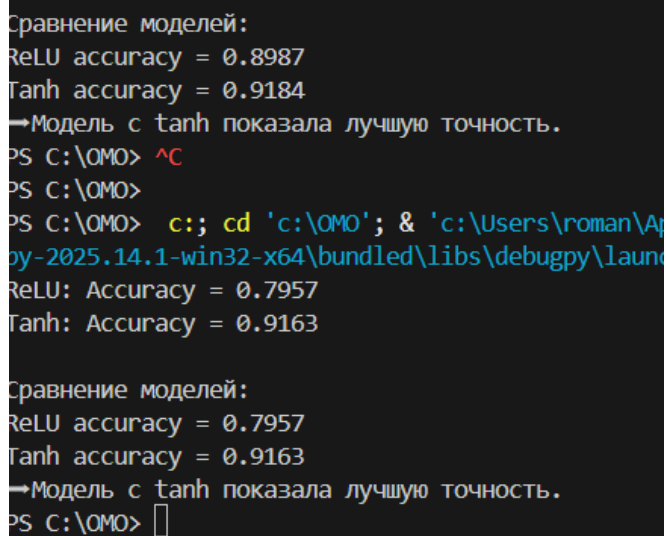
model.eval()
with torch.no_grad():
    y_pred_test = model(X_test_t)
    y_pred_labels = (y_pred_test > 0.5).float()
    acc = accuracy_score(y_test_t, y_pred_labels)
print(f"{name}: Accuracy = {acc:.4f}")
return acc

acc_relu = train_and_evaluate(nn.ReLU(), "ReLU")
acc_tanh = train_and_evaluate(nn.Tanh(), "Tanh")

print("\nСравнение моделей:")
print(f"ReLU accuracy = {acc_relu:.4f}")
print(f"Tanh accuracy = {acc_tanh:.4f}")

if acc_tanh > acc_relu:
    print("→ Модель с tanh показала лучшую точность.")
else:
    print("→ Модель с ReLU показала лучшую точность.")

```



```

Сравнение моделей:
ReLU accuracy = 0.8987
Tanh accuracy = 0.9184
→Модель с tanh показала лучшую точность.
PS C:\ОМО> ^C
PS C:\ОМО>
PS C:\ОМО> c.; cd 'c:\ОМО'; & 'c:\Users\roman\AppData\Local\Microsoft\WindowsApps\python-2025.14.1-win32-x64\bundled\libs\debugpy\launcher
ReLU: Accuracy = 0.7957
Tanh: Accuracy = 0.9163

Сравнение моделей:
ReLU accuracy = 0.7957
Tanh accuracy = 0.9163
→Модель с tanh показала лучшую точность.
PS C:\ОМО> 

```

Вывод: : построил, обучил и оценил многослойный перцептрон (MLP) для решения задачи классификации.