Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Системы искусственного интеллекта»

Лабораторная работа №4

«Основы нейронных сетей»

Вариант 15

Выполнила

Студентка группы ИВТАСбд-31

Микка Е. И.

Ульяновск, 2024

# Общее задание на лабораторную работу

В данной лабораторной работе необходимо:

1. Разделить исходную выборку на обучающую и тестовую выборки.
2. Произвести масштабирование признаков.
3. Обучить две модели нейронной сети (Perceptron и MLPClassifier) на обучающей выборке.
4. Проверить точность модели на тестовой выборке.
5. Провести эксперименты и определить наилучшие параметры: коэффициент обучения, параметр регуляризации, функцию оптимизации.

# Ход выполнения с описанием программной реализации

Задание 1.

Код загружает набор данных по варианту, разделяет его на обучающую и тестовую выборки (80/20), и преобразует их в одномерный формат для обучения модели машинного обучения.

phishing\_websites = fetch\_ucirepo(id=327)  
  
X = phishing\_websites.data.features  
Y = phishing\_websites.data.targets  
  
split\_index = int(0.8 \* len(X))  
X\_train, X\_test = X[:split\_index], X[split\_index:]  
Y\_train, Y\_test = Y[:split\_index], Y[split\_index:]  
Y\_train = Y\_train.values.ravel()  
Y\_test = Y\_test.values.ravel()

Задание 2.

Этот код использует StandardScaler для нормализации обучающего набора данных X\_train. Затем он применяет те же параметры нормализации к тестовому набору данных X\_test.

scaler = StandardScaler()  
  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)

Задание 3-4.

Код представляет собой процедуру обучения двух моделей машинного обучения: персептрона и многослойного перцептрона (MLP). Персептрон используется для простых задач классификации, а MLP - для более сложных. Обе модели обучаются на наборе данных, а затем их производительность оценивается на основе тестового набора данных для определения точности классификации.

#Персептрон  
  
# Создание и обучение персептрона  
perceptron = Perceptron(max\_iter=800)  
perceptron.fit(X\_train, Y\_train)  
  
# Предсказание и оценка точности  
Y\_pred = perceptron.predict(X\_test)  
accuracy = accuracy\_score(Y\_test, Y\_pred)  
print(f'Точность персептрона: {accuracy:.10f}')  
  
#MLP  
  
# Создание и обучение MLP классификатора  
mlp = MLPClassifier(max\_iter=800)  
  
mlp.fit(X\_train, Y\_train)  
  
# Предсказание и оценка точности  
y\_pred = mlp.predict(X\_test)  
accuracy = accuracy\_score(Y\_test, Y\_pred)  
print(f'Точность MLP классификатора: {accuracy:.10f}')

Задание 5.

Этот код выполняет подбор параметров для двух моделей машинного обучения — персептрона и многослойного перцептрона (MLP). Он итеративно обучает модели с различными значениями коэффициентов обучения, параметров регуляризации и функций оптимизации, чтобы найти наилучшие комбинации для каждой модели. После обучения моделей код оценивает их точность на тестовых данных и выводит наилучшие найденные значения параметров для каждой модели.

learning\_rates = [0.00001, 0.0001, 0.001, 0.01, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5]  
alpha = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0]  
fun\_opt = ['lbfgs', 'sgd', 'adam']  
  
lr\_mpl = []  
lr\_pr = []  
a\_mpl = []  
a\_pr = []  
opt\_mpl = []  
  
for lr in learning\_rates:  
 mlp = MLPClassifier(max\_iter=800, learning\_rate\_init=lr)  
 perceptron = Perceptron(max\_iter=800, eta0=lr)  
 mlp.fit(X\_train\_scaled, Y\_train)  
 perceptron.fit(X\_train\_scaled, Y\_train)  
 lr\_mpl.append(accuracy\_score(Y\_test, mlp.predict(X\_test\_scaled)))  
 lr\_pr.append(accuracy\_score(Y\_test, perceptron.predict(X\_test\_scaled)))  
best\_mpl = lr\_mpl.index(max(lr\_mpl))  
best\_pr = lr\_pr.index(max(lr\_pr))  
print(f'Наилучшие коэффиценты обучения:\n'  
 f'Перцептрон - {learning\_rates[best\_pr]}\n'  
 f'MLP - {learning\_rates[best\_mpl]}\n')  
  
for a in alpha:  
 mlp = MLPClassifier(max\_iter=800, alpha=a)  
 perceptron = Perceptron(max\_iter=800, alpha=a)  
 mlp.fit(X\_train\_scaled, Y\_train)  
 perceptron.fit(X\_train\_scaled, Y\_train)  
 a\_mpl.append(accuracy\_score(Y\_test, mlp.predict(X\_test\_scaled)))  
 a\_pr.append(accuracy\_score(Y\_test, perceptron.predict(X\_test\_scaled)))  
best\_mpl = a\_mpl.index(max(a\_mpl))  
best\_pr = a\_mpl.index(max(a\_mpl))  
print(f'Наилучшие параметр регуляризации:\n'  
 f'Перцептрон - {alpha[best\_pr]}\n'  
 f'MLP - {alpha[best\_mpl]}\n')  
  
for opt in fun\_opt:  
 mlp = MLPClassifier(max\_iter=800, solver=opt)  
 mlp.fit(X\_train\_scaled, Y\_train)  
 opt\_mpl.append(accuracy\_score(Y\_test, mlp.predict(X\_test\_scaled)))  
best\_mpl = opt\_mpl.index(max(opt\_mpl))  
print(f'Наилучшая функция оптимизации:\n'  
 f'MLP - {fun\_opt[best\_mpl]}\n')

# Тестирование

График зависимости точности от коэффициента обучения



График зависимости точности от параметра регуляризации

****

График зависимости точности от функции оптимизации



Вывод в консоли

