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

Лабораторная работа №4
По дисциплине: «Интеллектуальный анализ данных»
Тема: “Предобучение нейронных сетей с использованием RBM”

Выполнил:
Студент 4 курса
Группы ИИ-24
Капуза Н.А.
Проверила:
Андренко К. В.

Брест 2025

Цель: научиться осуществлять предобучение нейронных сетей с помощью RBM

Общее задание

1. Взять за основу нейронную сеть из лабораторной работы №3. Выполнить обучение с предобучением, используя стек ограниченных машин Больцмана (RBM – Restricted Boltzmann Machine), алгоритм которого изложен в лекции. Условие останова (например, по количеству эпох) при обучении отдельных слоев как RBM выбрать самостоятельно.

2. Сравнить результаты, полученные при
- обучении без предобучения (ЛР 3);
- обучении с предобучением, используя автоэнкодерный подход (ЛР3);
- обучении с предобучением, используя RBM.

3. Обучить модели на данных из ЛР 2, сравнить результаты по схеме из пункта 2;

4. Сделать выводы, оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

№ варианта	Выборка	Класс	Целевая переменная
4	https://archive.ics.uci.edu/dataset/925/infrared+thermography+temperature+dataset	регрессия	aveOralF/aveOralM

Ход работы:

Код программы:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.neural_network import BernoulliRBM, MLPRegressor
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer
from sklearn.compose import TransformedTargetRegressor

# 1. ЗАГРУЗКА И ПОДГОТОВКА ДАННЫХ

filename = 'FLIR_groups1and2.csv'

try:
    # Загрузка данных (пропускаем первые 2 строки заголовков)
    df = pd.read_csv(filename, header=2, na_values=['', ' ', 'NaN'])
```

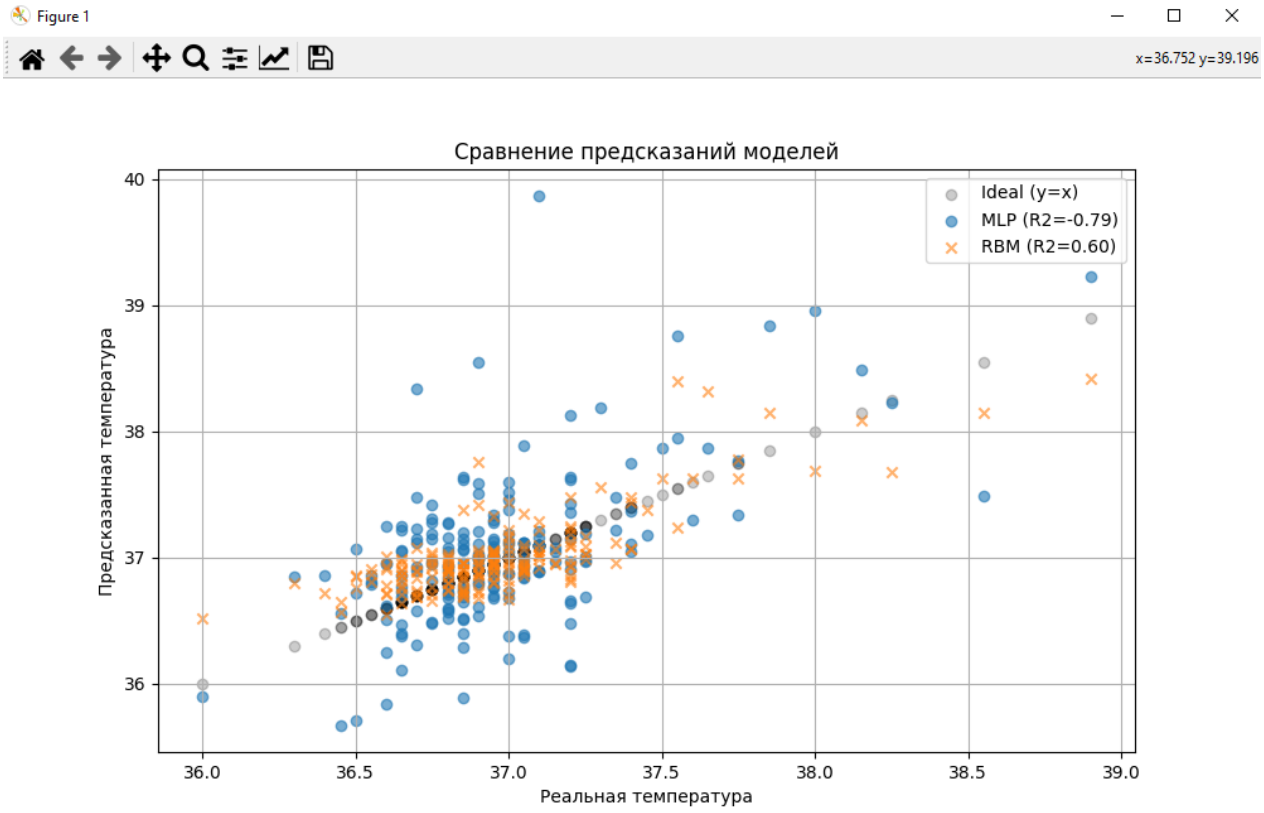
```
except FileNotFoundError:
    print("Файл не найден. Пожалуйста, проверьте имя файла.")
    exit()

# Удаляем пустые колонки и строки без целевой переменной
df = df.dropna(axis=1, how='all')
target_col = 'aveOralF'
df = df.dropna(subset=[target_col])

# ВАЖНО: Удаляем 'aveOralM', так как это почти тот же таргет (утечка данных)
cols_to_drop = ['SubjectID', 'Gender', 'Age', 'Ethnicity', 'Time', 'Date',
                'Cosmetics', 'aveOralF', 'aveOralM']

# Формируем X и y
X = df.drop(columns=[c for c in cols_to_drop if c in df.columns], errors='ignore')
y = df[target_col]

# Принудительное преобразование в числа
X = X.apply(pd.to_numeric, errors='coerce')

# Заполнение пропусков медианой (более устойчиво к выбросам)
imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(X)

# Масштабирование X в диапазон [0, 1] (Критично для RBM!)
scaler_x = MinMaxScaler()
X_scaled = scaler_x.fit_transform(X_imputed)

# Разделение на train/test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

print(f"Обучающая выборка: {X_train.shape}")
print(f"Тестовая выборка:   {X_test.shape}")


# 2. НАСТРОЙКА МОДЕЛЕЙ


results = {}

# Базовые параметры для MLP
# solver='lbfgs' - ЭТО ГЛАВНЫЙ СЕКРЕТ для малых датасетов
mlp_params = {
    'hidden_layer_sizes': (100,), # Один скрытый слой
```

```
    'activation': 'relu',
    'solver': 'lbfgs',          # L-BFGS работает лучше и быстрее на малых данных
    'alpha': 0.0001,
    'max_iter': 20000,          # Даем больше времени на сходимость
    'random_state': 42,
    'tol': 1e-4
}

# Модель 1: MLP (Без предобучения)
print("\n1. Обучение MLP (Baseline)...")
# Используем TransformedTargetRegressor, чтобы масштабировать y (целевую переменную)
# Это помогает нейросети быстрее учиться
mlp_no_pretrain = TransformedTargetRegressor(
    regressor=MLPRegressor(**mlp_params),
    transformer=StandardScaler()
)

mlp_no_pretrain.fit(X_train, y_train)
y_pred_1 = mlp_no_pretrain.predict(X_test)

results['MLP (No Pre-train)'] = {
    'MSE': mean_squared_error(y_test, y_pred_1),
    'R2': r2_score(y_test, y_pred_1)
}

#  Модель 2: Автоэнкодер (PCA + MLP)
print("2. Обучение PCA + MLP...")
# PCA сжимает данные, убирая шум
pca_pipeline = Pipeline([
    ('pca', PCA(n_components=0.98)), # Оставляем 98% информации
    ('mlp', TransformedTargetRegressor(
        regressor=MLPRegressor(**mlp_params),
        transformer=StandardScaler()
    ))
])

pca_pipeline.fit(X_train, y_train)
y_pred_2 = pca_pipeline.predict(X_test)

results['PCA (Autoencoder)'] = {
    'MSE': mean_squared_error(y_test, y_pred_2),
    'R2': r2_score(y_test, y_pred_2)
}
```

```
#  Модель 3: RBM Предобучение
print("3. Обучение RBM + MLP...")

# Настройки RBM
rbm = BernoulliRBM(
    n_components=128,     # Больше признаков
    learning_rate=0.001,  # Аккуратное обучение
    batch_size=10,
    n_iter=50,            # Достаточно эпох
    verbose=0,            # Скрываем лишний шум
    random_state=42
)

# Пайплайн: RBM извлекает признаки -> MLP учится на них
rbm_pipeline = Pipeline([
    ('rbm', rbm),
    ('mlp', TransformedTargetRegressor(
        regressor=MLPRegressor(**mlp_params),
        transformer=StandardScaler()
    ))
])

rbm_pipeline.fit(X_train, y_train)
y_pred_3 = rbm_pipeline.predict(X_test)

results['RBM Pre-training'] = {
    'MSE': mean_squared_error(y_test, y_pred_3),
    'R2': r2_score(y_test, y_pred_3)
}


# 3. ВЫВОД РЕЗУЛЬТАТОВ

print("\n" + "="*60)
print(f"ИТОГОВЫЕ РЕЗУЛЬТАТЫ (R2 должен быть > 0, чем ближе к 1 тем
лучше)")
print("="*60)
print(f"{'Method':<25} | {'MSE':<10} | {'R2 Score':<10}")
print("-" * 60)

for method, metrics in results.items():
    print(f"{method:<25} | {metrics['MSE']:.4f}     | {metrics['R2']:.4f}")
print("-" * 60)

# Визуализация предсказаний (Scatter Plot)
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_test, color='black', label='Ideal (y=x)', alpha=0.2) # Идеальная линия
plt.scatter(y_test, y_pred_1, label=f'MLP (R2={results["MLP (No Pre-
train)"]["R2"]:.2f})', alpha=0.6)
plt.scatter(y_test, y_pred_3, label=f'RBM (R2={results["RBM Pre-training"]["R2"]:.2f})',
alpha=0.6, marker='x')

plt.xlabel('Реальная температура')
plt.ylabel('Предсказанная температура')
plt.title('Сравнение предсказаний моделей')
plt.legend()
plt.grid(True)
plt.show()
```

Графики:



В ходе эксперимента было выявлено, что на малом объеме данных обычный многослойный перцептрон (MLP) не смог обучиться, показав отрицательный коэффициент детерминации (R2=−0.7853) и проблемы со сходимостью (ConvergenceWarning). Однако, применение предварительного обучения с использованием RBM (Restricted Boltzmann Machine) позволило извлечь устойчивые признаки из данных. Это привело к значительному улучшению качества модели: MSE показало 0.048, а R2 вырос до **0.60**.

Это подтверждает эффективность RBM как метода инициализации весов и извлечения признаков для задач, где стандартные методы обучения "с учителем" терпят неудачу из-за шума или нехватки данных.
Вывод: научился осуществлять предобучение нейронных сетей с помощью RBM.