`ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы»

Дисциплина «Системы искусственного интеллекта»

ОТЧЁТ

ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №7

«Нейросетевой апроксиматор функции 3-х переменных»

Выполнили студенты — Шефнер А.
Факультет: АИТ — Кот. Н.Д.
Группа: ИВБ-211 — Егупов Н.М.
Ахмедов Х.А.

Проверил: Пугачев С.В.

Санкт-Петербург

2025

Задание

Разработать нейросеть, аппроксимирующую сумму трёх переменных

Ход работы

Использованные библиотеки

In [1]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
```

Подготовка тренировочных данных

В качестве набора тренировочных данных выбраны случайные тройки чисел в диапазоне $[0; 1]$ (входной сигнал) и их суммы в диапазоне $[0; 3]$ (выходной сигнал). Примеров сгенерировано 50000

In [2]:
```
SEED = 69420
np.random.seed(SEED)
torch.manual_seed(SEED)

inputs = np.random.rand(50000, 3).astype(np.float32)
targets = np.array(np.sum(inputs, axis=-1))

X_train = torch.tensor(inputs)
y_train = torch.tensor(targets).reshape(-1, 1)
```

Определение модели

Для данной работы составлена модель из трёх слоёв

In [3]:
```
class Baseline(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Linear(3, 64)
        self.layer2 = nn.Linear(64, 32)
        self.layer3 = nn.Linear(32, 1)

    def forward(self, x):
        x = torch.relu(self.layer1(x))
        x = torch.relu(self.layer2(x))
        x = self.layer3(x)
        return x
```

```
model = Baseline()
model
```

Out[3]:
```
Baseline(
  (layer1): Linear(in_features=3, out_features=64, bias=True)
  (layer2): Linear(in_features=64, out_features=32, bias=True)
  (layer3): Linear(in_features=32, out_features=1, bias=True)
)
```

Обучение модели

In [4]:
```
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.MSELoss()

model.train()
n_epochs = 1000
for epoch in range(n_epochs):
    optimizer.zero_grad()
    outputs = model(X_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)  # Gradient c
    optimizer.step()

    if (epoch + 1) % 100 == 0:
        print(f'Epoch [{epoch+1}/{n_epochs}], Loss: {loss.item():.6f}')
```

```
Epoch [100/1000], Loss: 0.016611
Epoch [200/1000], Loss: 0.000554
Epoch [300/1000], Loss: 0.000324
Epoch [400/1000], Loss: 0.000234
Epoch [500/1000], Loss: 0.000183
Epoch [600/1000], Loss: 0.000150
Epoch [700/1000], Loss: 0.000126
Epoch [800/1000], Loss: 0.000108
Epoch [900/1000], Loss: 0.000097
Epoch [1000/1000], Loss: 0.000088
```

Проверка производительности обученной модели

In [5]:
```
model.eval()

test_data = [
    [0.2 , 0.3, 0.91],
    [0.11, 0.9, 0.4 ],
    [0.65, 0.0, 0.2 ],
    [0.8 , 0.9, 1.0 ],
    [1.0 , 1.0, 1.0 ],
    [0.0 , 0.0, 0.0 ],
]

predictions = []
with torch.no_grad():
    predictions = model(torch.tensor(test_data))

data = [
```

```
    [*input, sum(input), float(output[0]), abs(sum(input) - float(output[
    for input, output in zip(test_data, predictions)
]

pd.DataFrame(
    data,
    columns = ["A", "B", "C", "A+B+C", "Prediction", "Error"]
)
```

Out[5]:

	A	B	C	A+B+C	Prediction	Error
0	0.20	0.3	0.91	1.41	1.415028	0.005028
1	0.11	0.9	0.40	1.41	1.411467	0.001467
2	0.65	0.0	0.20	0.85	0.846405	0.003595
3	0.80	0.9	1.00	2.70	2.685898	0.014102
4	1.00	1.0	1.00	3.00	2.976923	0.023077
5	0.00	0.0	0.00	0.00	0.122521	0.122521

Вывод

В ходе данной работы была разработана нейросеть, апроксимирующая функцию суммы трёх переменных. Интересный результат исследования - разное поведение в крайних значениях выходного диапазона. Для значений $(0, 0, 0)$ точность оказалась на порядок ниже, чем для $(1, 1, 1)$