Лабораторная работа № 2

Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Цель работы: исследование свойств линейной нейронной сети и алгоритмов ее обучения, применение сети в задачах аппроксимации и фильтрации

Студент	Мариничев И.А.
Группа	М8О-408Б-19
Вариант	5

Определим входные данные для задачи аппроксимации функции

```
D = 5  # число дискрет


def x(t):
    return np.sin(t**2 - 7 * t)


t = np.arange(0, 5, 0.025)  # отрезок с шагом
X = x(t).tolist()           # временная последовательность


sequences = [X[i:i+D] for i in range(0, len(X) - D)]    # последовательности
из D дискрет
upcoming_points = [X[i] for i in range(D, len(X))]      # последующие
дискреты
```

Создадим класс линейной нейронной сети, выполняющей перемножение матрицы весов и матрицы входа с прибавлением вектора смещения

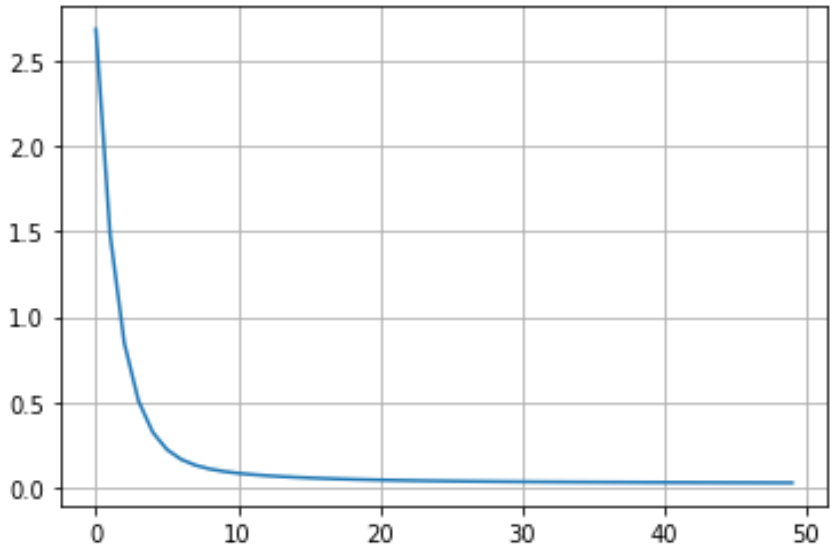
```
class ADALINE(nn.Module):
    def __init__(self, in_features: int, out_features: int, bias: bool =
True):
        super().__init__()
        self.weights = nn.Parameter(torch.randn(in_features, out_features))
        self.bias = bias
        if bias:
            self.bias_term = nn.Parameter(torch.randn(out_features))

    def forward(self, x):
        x = x @ self.weights
        if self.bias:
            x += self.bias_term
        return x
```

Наша нейросеть будет принимать на вход `D` дискрет и иметь один выходной нейрон, последущую дискрету.

В качестве функции потерь берем `nn.MSELoss()`

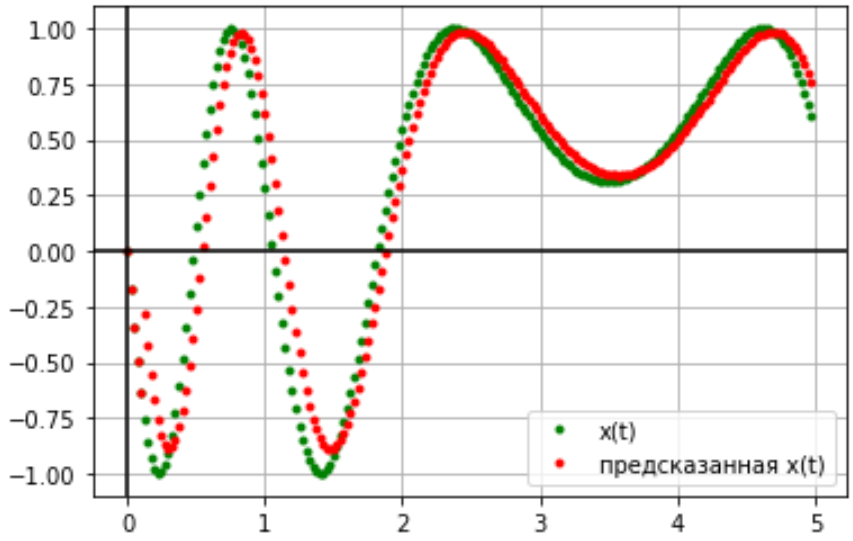
Обучим модель

Посмотрим на график функции потерь, вычисляющей `MSE` между исходными и полученными данными

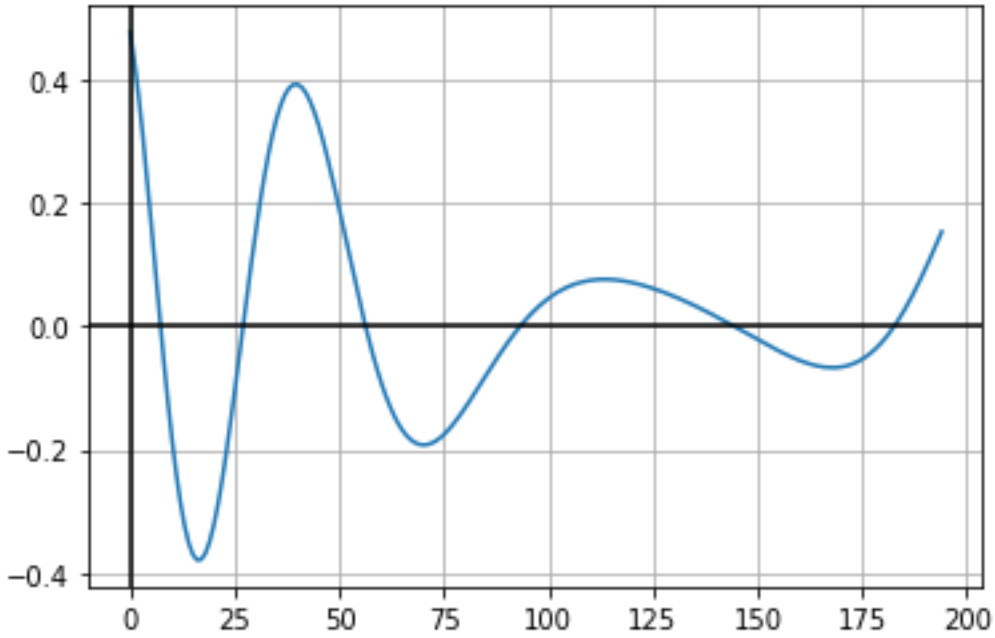


Теперь соберем предсказания модели для каждой последовательности из D дискрет и заодно посчитаем отклонения от истинных значений

И для сравнения посмотрим на исходную и предсказанную временные последовательности



И визуализируем наши отклонения



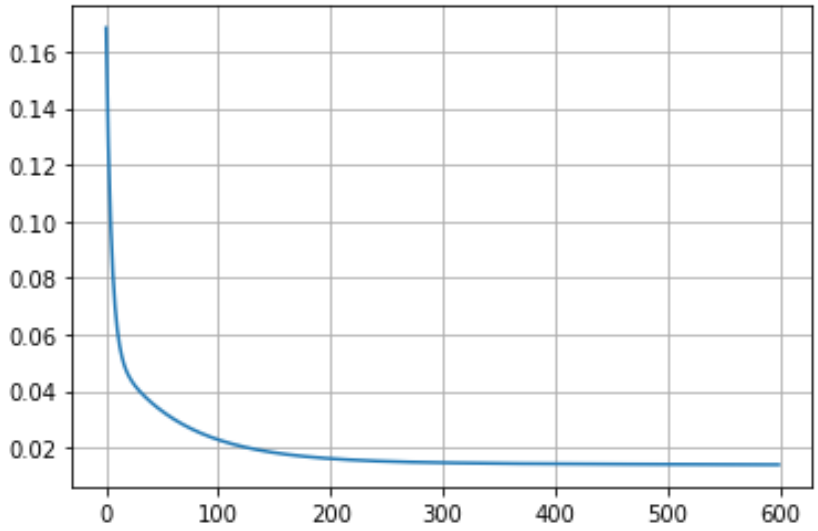
Определим входные данные для задачи подавления помех

```
D = 4  # число дискрет


def true_signal(t):
    return np.sin(t**2 - 6 * t + 3)


def noized_signal(t):
    return (1 / 4) * np.sin(t**2 - 6 * t - 2 * np.pi)


t = np.arange(0, 5, 0.025)      # отрезок с шагом
X = true_signal(t).tolist()     # истинный сигнал
Y = noized_signal(t).tolist()   # зашумленный сигнал


noized_sequences = [Y[i:i+D] for i in range(0, len(Y) - D)] #
последовательности из D дискрет (зашумленный сигнал)
upcoming_points_true = [X[i] for i in range(D, len(X))]     # последующие
дискреты (истинный сигнал)
```

Наша вторая нейросеть будет принимать на вход D зашумленных дискрет и иметь один выходной нейрон, последущую очищенную дискрету

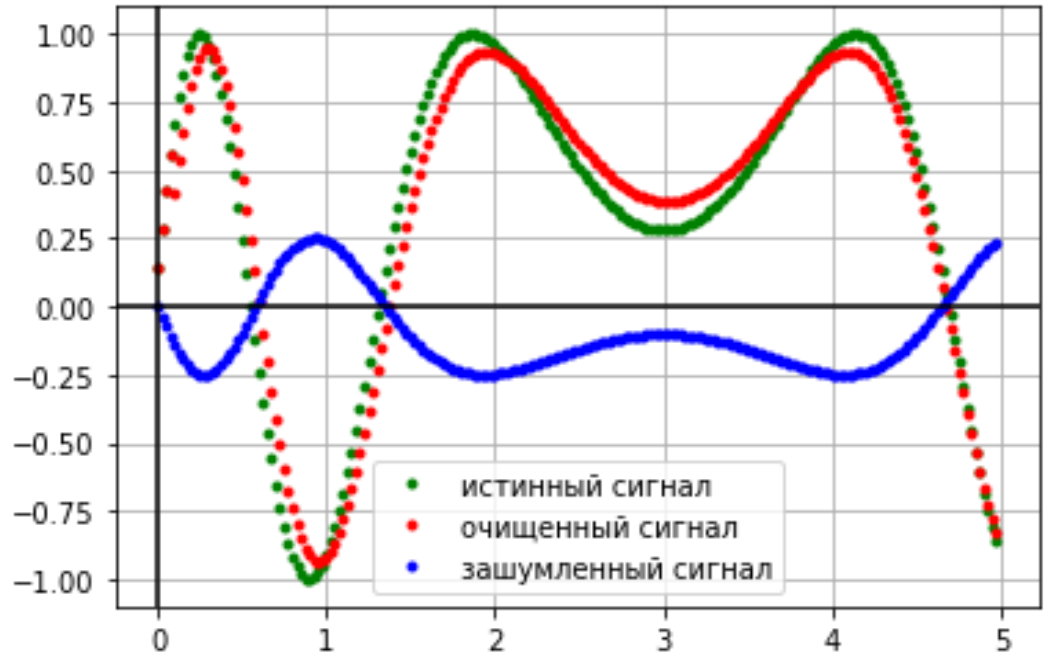
Обучим модель

Посмотрим на график функции потерь, вычисляющей MSE между исходными и полученными данными

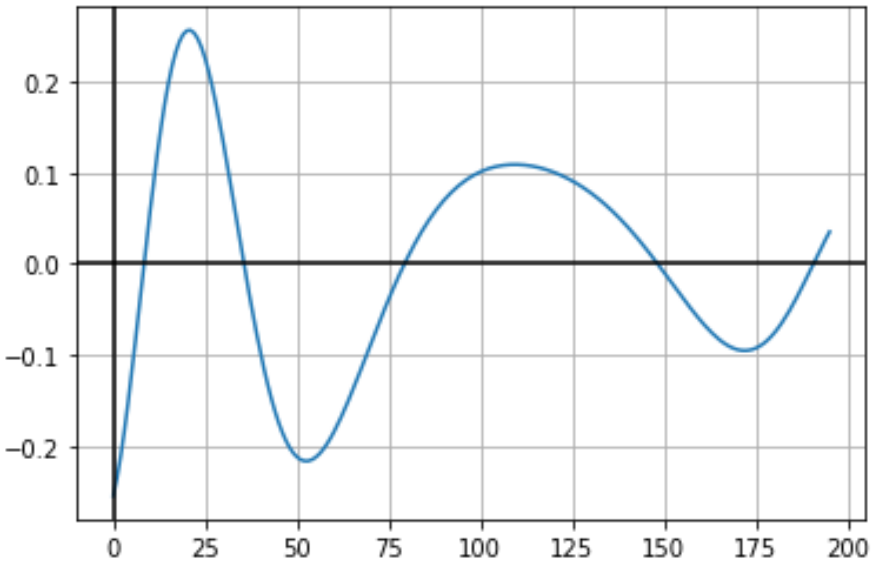


Теперь соберем предсказания модели для каждой последовательности из D дискрет и заодно посчитаем отклонения от истинных значений

И теперь посмотрим в сравнении на истинный, зашумленный и очищенный сигналы



И визуализируем наши отклонения



Выводы: в ходе данной работы была построена линейная нейросетевая модель, которая была применена для двух задач:

- аппроксимация функции
- подавление помех в сигнале

И после обучения (50 и 600 эпох соответственно) были получены хорошие результаты