

**Московский авиационный институт  
(национальный исследовательский университет)**

**Институт №8 «Информационные технологии и прикладная  
математика»**

**Кафедра 806 «Вычислительная математика и  
программирование»**

**Лабораторная работа №2 по курсу «Нейроинформатика»**

Студент: В. В. Бирюков  
Преподаватель: И. А. Рожлейс  
Группа: М8О-407Б-19  
Дата:  
Оценка:  
Подпись:

**Москва, 2022**

## Лабораторная работа №2

**Тема:** Линейная нейронная сеть. Правило обучения Уидроу-Хоффа.

**Цель работы:** Исследование свойств линейной нейронной сети и алгоритмов её обучения, применения сети в задачах аппроксимации и фильтрации.

**Основные этапы работы:**

1. Использовать линейную нейронную сеть с задержками для аппроксимации функции. В качестве метода обучения использовать адаптацию.
2. Использовать линейную нейронную сеть в качестве адаптивного фильтра для подавления помех. Для настройки весовых коэффициентов использовать метод наименьших квадратов.

**Вариант:** 9

$$x = \sin(t^2 - 2t + 3), \quad t \in [0, 6], \quad h = 0.025$$

$$y = \sin(t^2 - 2t), \quad t \in [0, 6], \quad h = 0.025$$

$$x = \sin(t^2 - 2t + 3)$$

# 1 Исходный код

## Лабораторная работа № 2

Вариант: 9

```
[1]: import numpy as np
      from tensorflow import keras
      import matplotlib.pyplot as plt
```

```
[2]: %matplotlib inline
      import matplotlib_inline
      matplotlib_inline.backend_inline.set_matplotlib_formats('svg', 'pdf')
```

```
[3]: def plot_history(h, *metrics):
      for metric in metrics:
          print(f"{metric}: {h.history[metric][-1]:.4f}")
      figure = plt.figure(figsize=(5.5 * len(metrics), 3.5))
      for i, metric in enumerate(metrics, 1):
          ax = figure.add_subplot(1, len(metrics), i)
          ax.xaxis.get_major_locator().set_params(integer=True)
          plt.title(metric)
          plt.plot(h.history[metric], '-')
      plt.show()
```

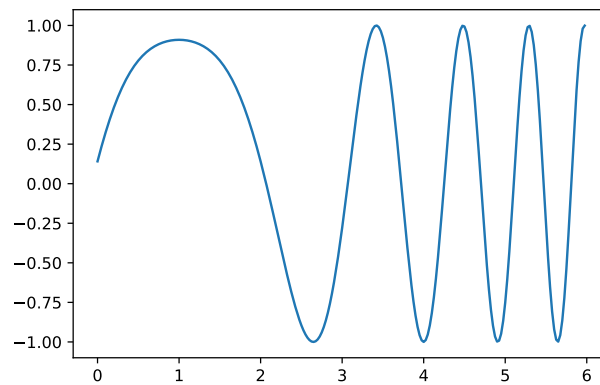
```
[4]: window = 5
```

## Аппроксимация функции

```
[5]: signal1 = lambda t: np.sin(t ** 2 - 2 * t + 3)
      t1 = np.arange(0, 6, 0.025)

      data1 = signal1(t1)
      target1 = data1[window:]
      data1 = np.array([data1[i:i+window] for i in range(0, len(data1) - window)])
```

```
[6]: plt.plot(t1, signal1(t1))
      plt.show()
```



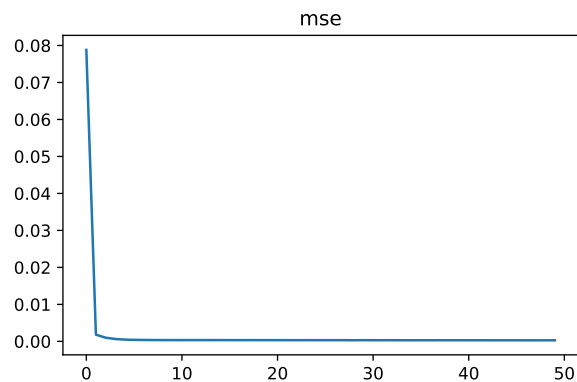
```
[7]: model1 = keras.models.Sequential([
      keras.layers.Dense(1, input_dim=window, activation='linear')
    ])

model1.compile(keras.optimizers.SGD(0.01), 'mse', ['mse'])

hist1 = model1.fit(data1, target1, batch_size=1, epochs=50, verbose=0, shuffle=True)
```

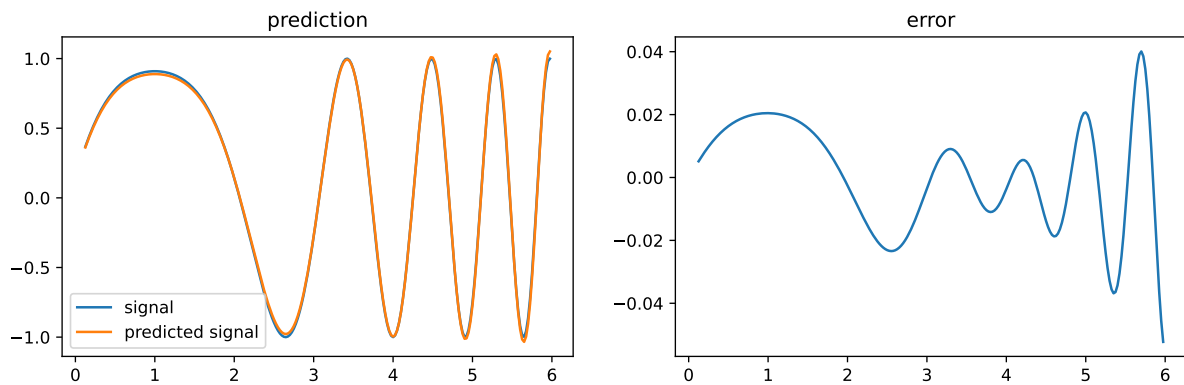
```
[8]: plot_history(hist1, 'mse')
```

mse: 0.0003



```
[9]: pred1 = model1.predict(data1)
figure = plt.figure(figsize=(12, 3.5))
figure.add_subplot(1, 2, 1)
plt.title('prediction')
plt.plot(t1[window:], target1, label='signal')
plt.plot(t1[window:], pred1, label='predicted signal')
```

```
plt.legend()
figure.add_subplot(1, 2, 2)
plt.title('error')
plt.plot(t1[window:], target1 - pred1.flat)
plt.show()
```

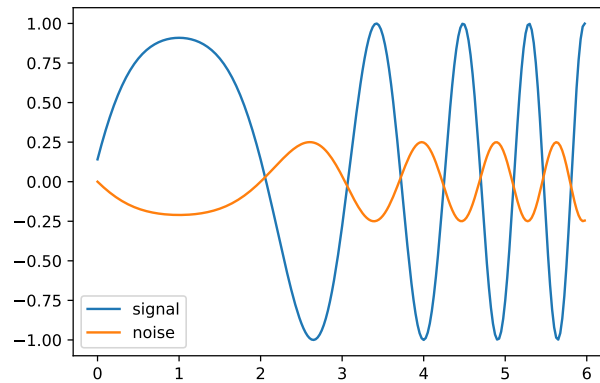


## Подавление помех

```
[10]: noise_signal = lambda t: np.sin(t ** 2 - 2 * t) / 4
signal2 = lambda t: np.sin(t ** 2 - 2 * t + 3)
t2 = np.arange(0, 6, 0.025)

data2 = noise_signal(t2)
data2 = np.array([data2[i:i+window] for i in range(0, len(data2) - window)])
target2 = signal2(t2)[window:]
```

```
[11]: plt.plot(t2, signal2(t2), label='signal')
plt.plot(t2, noise_signal(t2), label='noise')
plt.legend()
plt.show()
```



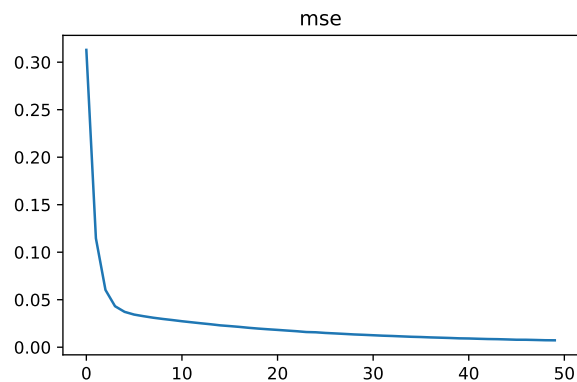
```
[12]: model2 = keras.models.Sequential([
        keras.layers.Dense(1, input_dim=window, activation='linear')
    ])

model2.compile(keras.optimizers.SGD(0.01), 'mse', ['mse'])

hist2 = model2.fit(data2, target2, batch_size=1, epochs=50, verbose=0, shuffle=True)
```

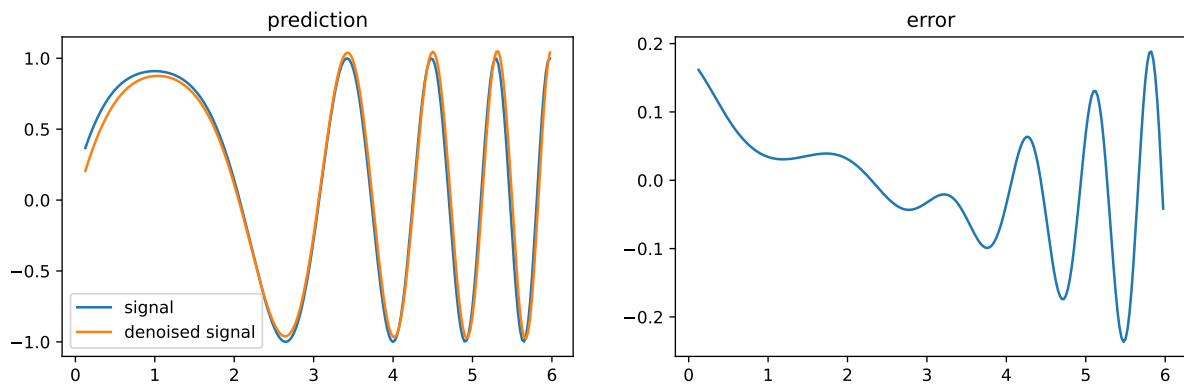
```
[13]: plot_history(hist2, 'mse')
```

mse: 0.0073



```
[14]: pred2 = model2.predict(data2)
figure = plt.figure(figsize=(12, 3.5))
figure.add_subplot(1, 2, 1)
plt.title('prediction')
plt.plot(t2[window:], target2, label='signal')
plt.plot(t2[window:], pred2, label='denoised signal')
```

```
plt.legend()
figure.add_subplot(1, 2, 2)
plt.title('error')
plt.plot(t2>window:], target2 - pred2.flat)
plt.show()
```



## 2 Выводы

В ходе выполнения лабораторной работы я познакомился с задачами аппроксимации и фильтрации функции и применил однослойную нейронную сеть для решения этих задач.