Московский Государственный Технический Университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»


# Отчёт по лабораторной работе № 5 по дисциплине «Проектирование интеллектуальных систем»

Исполнитель:
Саврасов П.А.
Группа ИУ5-24М
«__» _____ 2021 г.

Преподаватель:
Терехов В.И.
_____
«__» _____ 2021 г.

Москва, 2021 г.

# 1. Цель работы

Научиться работать с реккурентными нейронными сетями (RNN) в Tensorflow.

# 2. Упражнение 1

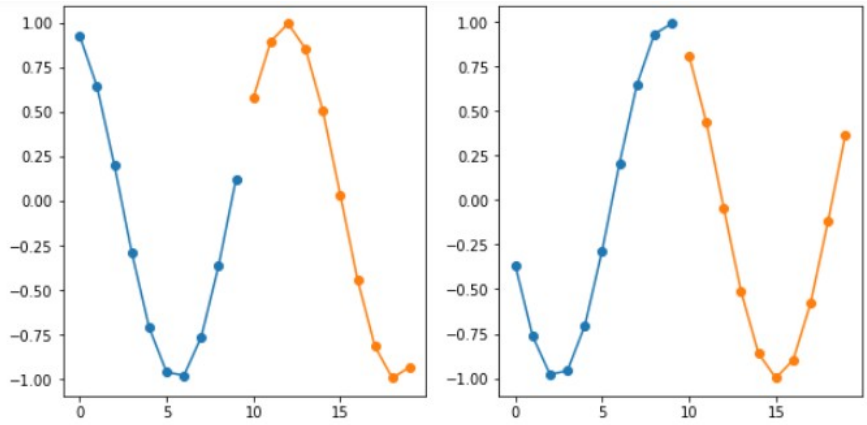
```
import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt

import seaborn as sns
import numpy as np
import random
import math
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

**Упражнение 1**

Генератор данных для упражнения 1 (изменённый)

```
def generate_x_y_data_v1(seq_length):
    batch_x = []
    batch_y = []

    rand = random.random() * 2 * math.pi
    sig1 = np.sin(
        np.linspace(
            0.0 * math.pi + rand,
            3.0 * math.pi + rand,
            seq_length * 2
        )
    )

    sig2 = np.cos(
        np.linspace(0.0 * math.pi + rand,
                    3.0 * math.pi + rand,
                    seq_length * 2
                   )
    )

    x1 = sig1[:seq_length]
    y1 = sig1[seq_length:]
    x2 = sig2[:seq_length]
    y2 = sig2[seq_length:]

    x_ = np.array([x1, x2])
    y_ = np.array([y1, y2])
    x_, y_ = x_.T, y_.T

    batch_x.append(x_)
    batch_y.append(y_)

    batch_x = np.array(batch_x)
    batch_y = np.array(batch_y)
    batch_x = np.array(batch_x).transpose((1, 0, 2))
    batch_y = np.array(batch_y).transpose((1, 0, 2))
    return batch_x, batch_y
```

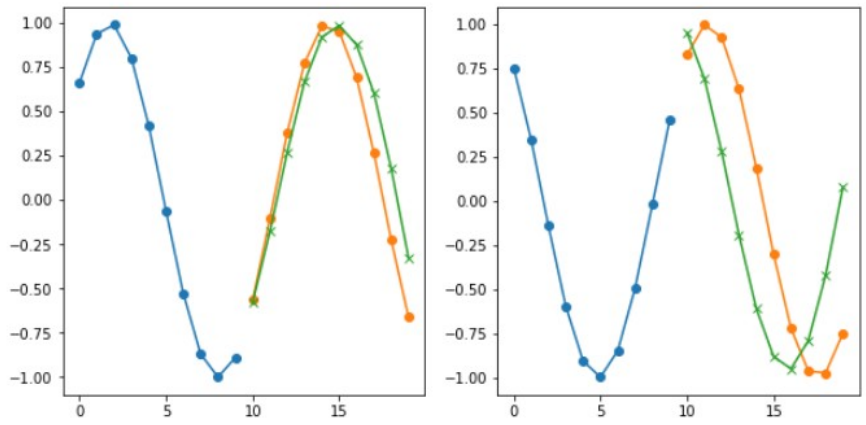
```
seqLen = 10
xTrain, yTrain = generate_x_y_data_v1(seqLen)
print('Формат выходных данных x:', xTrain.shape)
print('Формат выходных данных y:', yTrain.shape)

plt.subplots(figsize=(10,5))
plt.subplot(1, 2, 1)
plt.plot(range(seqLen), xTrain.T.reshape((2, seqLen))[0], 'o-')
plt.plot(range(seqLen, seqLen * 2), yTrain.T.reshape((2, seqLen))[0], 'o-')

plt.subplot(1, 2, 2)
plt.plot(range(seqLen), xTrain.T.reshape((2, seqLen))[1], 'o-')
plt.plot(range(seqLen, seqLen*2), yTrain.T.reshape((2, seqLen))[1], 'o-')
```



```
model = keras.Sequential()
model.add(layers.GRU(units=200, return_sequences=True, input_shape=(seqLen, 2)))
model.add(layers.GRU(units=100))
model.add(layers.Dense(units=2, activation='tanh'))

model.compile(
    optimizer = keras.optimizers.RMSprop(learning_rate=0.01),
    loss = keras.losses.MeanAbsoluteError()
)
for i in range(50):
    xTrain, yTrain = generate_x_y_data_v1(seqLen)
    model.fit(xTrain, yTrain, verbose=0)
yPred = model.predict(xTrain)
```

```
plt.subplots(figsize=(10,5))
plt.subplot(1, 2, 1)
plt.plot(range(seqLen), xTrain.T.reshape((2, seqLen))[0], 'o-')
plt.plot(range(seqLen, seqLen * 2), yTrain.T.reshape((2, seqLen))[0], 'o-')
plt.plot(range(seqLen, seqLen * 2), yPred.T[0], 'x-')

plt.subplot(1, 2, 2)
plt.plot(range(seqLen), xTrain.T.reshape((2, seqLen))[1], 'o-')
plt.plot(range(seqLen, seqLen * 2), yTrain.T.reshape((2, seqLen))[1], 'o-')
plt.plot(range(seqLen, seqLen * 2), yPred.T[1], 'x-')
```

```
[<matplotlib.lines.Line2D at 0x7fa04c2adf40>]
```



# 3. Упражнение 2

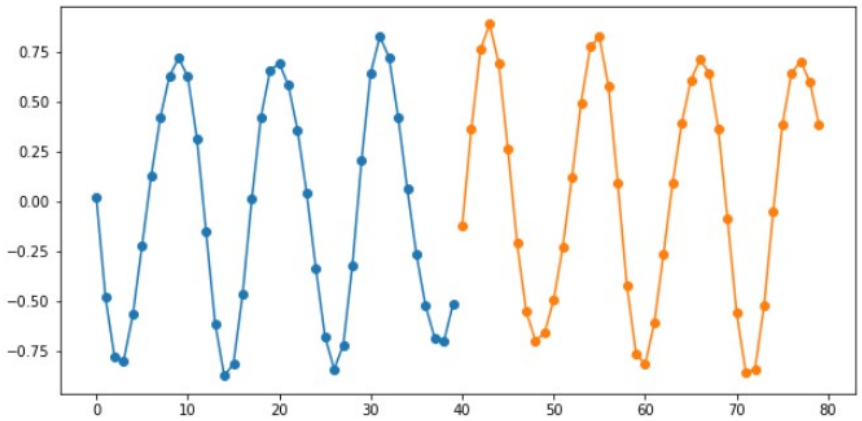
**Упражнение 2**

Генератор данных для упражнения 2 (изменённый)

```
def generate_x_y_data_v2(seq_length):
    batch_x = []
    batch_y = []

    offset_rand = random.random() * 2 * math.pi
    freq_rand = (random.random() - 0.5) / 1.5 * 15 + 0.5
    amp_rand = random.random() + 0.1
    sig1 = amp_rand * np.sin(
        np.linspace(
            seq_length / 15.0 * freq_rand * 0.0 * math.pi + offset_rand,
            seq_length / 15.0 * freq_rand * 3.0 * math.pi + offset_rand,
            seq_length * 2
        )
    )

    offset_rand = random.random() * 2 * math.pi
    freq_rand = (random.random() - 0.5) / 1.5 * 15 + 0.5
    amp_rand = random . random () * 1.2
    sig1 = amp_rand * np.cos(
        np.linspace(
            seq_length / 15.0 * freq_rand * 0.0 * math.pi + offset_rand,
            seq_length / 15.0 * freq_rand * 3.0 * math.pi + offset_rand,
            seq_length * 2
        )
    ) + sig1

    x1 = sig1[:seq_length]
    y1 = sig1[seq_length:]
    x_ = np.array([x1])
    y_ = np.array([y1])
    x_, y_ = x_.T, y_.T

    batch_x.append(x_)
    batch_y.append(y_)
    batch_x = np.array(batch_x)
    batch_y = np.array(batch_y)

    batch_x = np.array(batch_x).transpose((1, 0, 2))
    batch_y = np.array(batch_y).transpose((1, 0, 2))
    return batch_x, batch_y
```

```
seqLen = 40
x, y = generate_x_y_data_v2(seqLen)
print('Формат выходных данных x:', x.shape)
print('Формат выходных данных y:', y.shape)
x = x.reshape((seqLen))
y = y.reshape((seqLen))

plt.subplots(figsize=(10,5))
plt.plot(range(seqLen), x, 'o-')
plt.plot(range(seqLen, seqLen * 2), y, 'o-')
xTrain, yTrain = generate_x_y_data_v2(seqLen)
```

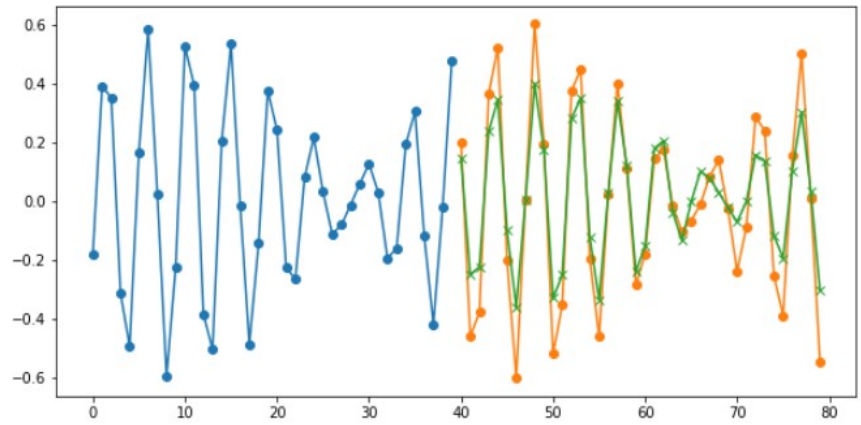
```
Формат выходных данных x: (40, 1, 1)
Формат выходных данных y: (40, 1, 1)
```



```
model = keras.Sequential()
model.add(layers.GRU(units=200, return_sequences=True, input_shape=(seqLen, 1)))
model.add(layers.GRU(units=100))
model.add(layers.Dense(units=1, activation='tanh'))

model.compile(
    optimizer = keras.optimizers.RMSprop(learning_rate=0.01),
    loss = keras.losses.MeanAbsoluteError()
)
for i in range(10):
    xTrain, yTrain = generate_x_y_data_v2(seqLen)
    model.fit(xTrain, yTrain, epochs=5, verbose=0)
yPred = model.predict(xTrain)
```

```
plt.subplots(figsize=(10,5))

plt.plot(range(seqLen), xTrain.T.reshape((1, seqLen))[0], 'o-')
plt.plot(range(seqLen, seqLen * 2), yTrain.reshape(seqLen), 'o-')
plt.plot(range(seqLen, seqLen * 2), yPred, 'x-')
```

```
[<matplotlib.lines.Line2D at 0x7fa001136f10>]
```




# 4. Упражнение 3

**Упражнение 3**

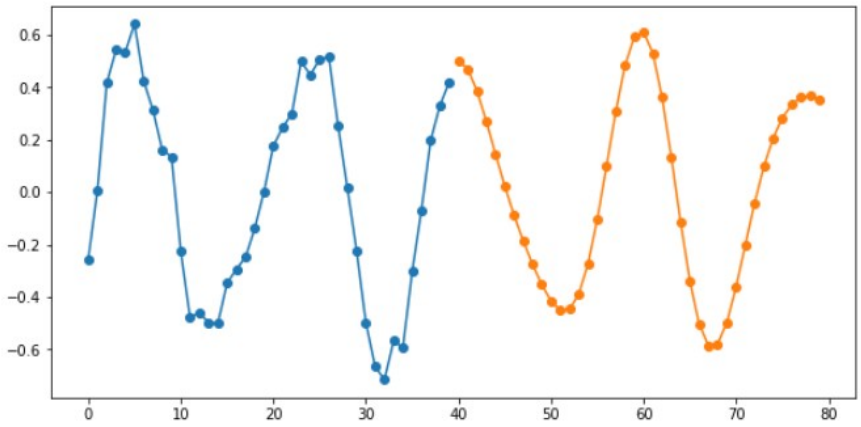
Генератор данных для упражнения 3 (изменённый)

```
def generate_x_y_data_v3 (seq_length):
    x, y = generate_x_y_data_v2(seq_length )
    noise_amount = random.random() * 0.15 + 0.10
    x = x + noise_amount * np.random.randn(seq_length, 1, 1)
    avg = np.average(x)
    std = np.std(x) + 0.0001
    x = x - avg
    y = y - avg
    x = x / std / 2.5
    y = y / std / 2.5
    return x, y
```

```
seqLen = 40
x, y = generate_x_y_data_v3(seqLen)
print('Формат выходных данных x:', x.shape)
print('Формат выходных данных y:', y.shape)
x = x.reshape((seqLen))
y = y.reshape((seqLen))

plt.subplots(figsize=(10,5))
plt.plot(range(seqLen), x, 'o-')
plt.plot(range(seqLen, seqLen * 2), y, 'o-')
```

```
Формат выходных данных x: (40, 1, 1)
Формат выходных данных y: (40, 1, 1)
```

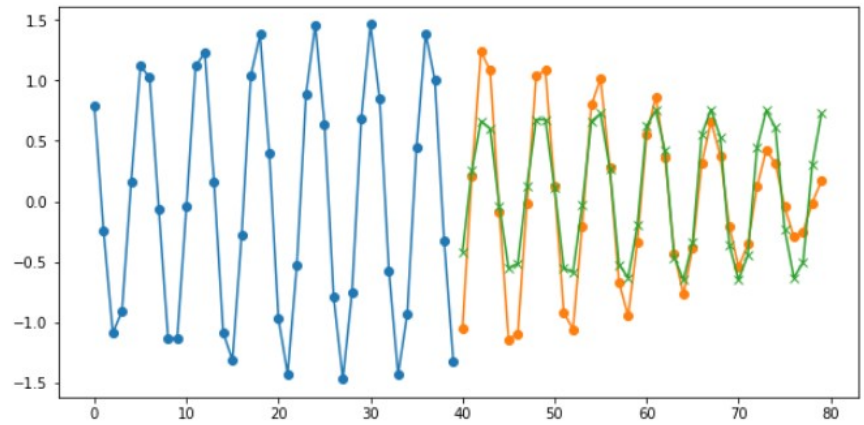
```
[<matplotlib.lines.Line2D at 0x7fa000db51f0>]
```



```
model = keras.Sequential()
model.add(layers.GRU(units=200, return_sequences=True, input_shape=(seqLen, 1)))
model.add(layers.GRU(units=100))
model.add(layers.Dense(units=1, activation='tanh'))

model.compile(
    optimizer = keras.optimizers.RMSprop(learning_rate=0.01),
    loss = keras.losses.MeanAbsoluteError()
)
for i in range(10):
    xTrain, yTrain = generate_x_y_data_v2(seqLen)
    model.fit(xTrain, yTrain, epochs=5, verbose=0)
yPred = model.predict(xTrain)
```

```
plt.subplots(figsize=(10,5))

plt.plot(range(seqLen), xTrain.T.reshape((1, seqLen))[0], 'o-')
plt.plot(range(seqLen, seqLen * 2), yTrain.reshape(seqLen), 'o-')
plt.plot(range(seqLen, seqLen * 2), yPred, 'x-')
```

```
[<matplotlib.lines.Line2D at 0x7f9fd9fd2910>]
```



## 5. Выводы по работе:

Научился работать с реккурентными нейронными сетями (RNN) в Tensorflow.

## 6. Контрольные вопросы:

**6.1. В чем преимущество рекуррентных нейронных сетей по сравнению с обычными перцептронами?**

Рекуррентные сети имеют «память» в отличие от перцептронами сетями, так как есть «память» это дает возможность анализировать данные, где порядок следования важен.

**6.2. Что такое регуляризация и зачем она нужна?**

Регуляризация в статистике, машинном обучении, теории обратных задач — метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу или предотвратить переобучение.

**6.3. Что такое пакетный, мини-пакетный и онлайновый градиентный спуск?**

Пакетный градиентный спуск — это один из видов алгоритма градиентного спуска. Его особенность состоит в том, что он вычисляет ошибку для каждого примера в наборе обучающих данных.

Мини-пакетный градиентный спуск – это один из видов алгоритма градиентного спуска, которые используются для расчета коэффициентов модели.

Онлайновый градиентный спуск – Градиентный спуск(подъем) - один из наиболее популярных методов оптимизации в машинном обучении. Оптимизация - это процесс нахождения точек максимума/минимума некоторой функции