МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

ОТЧЕТ

Лабораторная работа №6

по курсу «Проектирование интеллектуальных систем»

Тема: «Использование рекуррентных нейронных сетей для предсказания временных рядов»

ИСПОЛНИТЕЛЬ: группа ИУ5-22

__Петрова Ирина___
ФИО

_____
подпись

"25"___05____2020 г.

ПРЕПОДАВАТЕЛЬ: _____
ФИО

_____
подпись

"__"_____2020 г.

Москва - 2020

Задание

На основе приведенного кода нужно выполнить 3 упражнения. Для успешного запуска нужно структурировать код следующим образом:

1. Генерация данных (процедуры генерации данных приведены в приложении)
2. Определение параметров нейросети
3. Описание графа вычислений
4. Описание функций потерь и оптимизатора
5. Обучение и валидация нейронной сети
6. Визуализация результатов (динамика ошибки на обучающей и тестовой выборке по итерациям и визуализация предсказаний)

Нейронная сеть имеет набор настраиваемых параметров. Они приведены в приложении 2

Процедуры генерации данных для каждого из упражнений приведены в приложении 3.

В упражнении 1 имеется 2 временных ряда, которые связаны вместе. Это также можно интерпретировать как двумерный временной ряд. Для выполнения упражнения 1 требуется просто собрать код из приведенных частей и запустить его.

В упражнении 2 имеется 1 сигнал, который является суперпозицией двух синусоид с разным периодом и смещением. Чтобы получить корректные предсказания, необходимо изменить гиперпараметры нейронной сети.

Упражнение 3 аналогично упражнению 2, однако входные данные зашумлены. Здесь также нужно будет настроить гиперпараметры нейронной сети, чтобы получить корректные предсказания.

В отчете для каждого упражнения необходимо привести используемые гиперпараметры, лог со значениями функции потерь на этапе обучения, а также графики изменения ошибки в процессе обучения и графики полученных предсказаний.

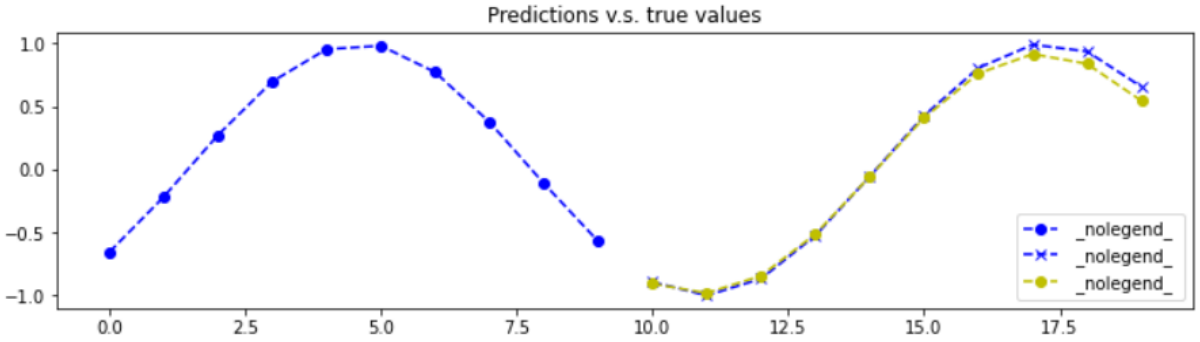
Выполнение

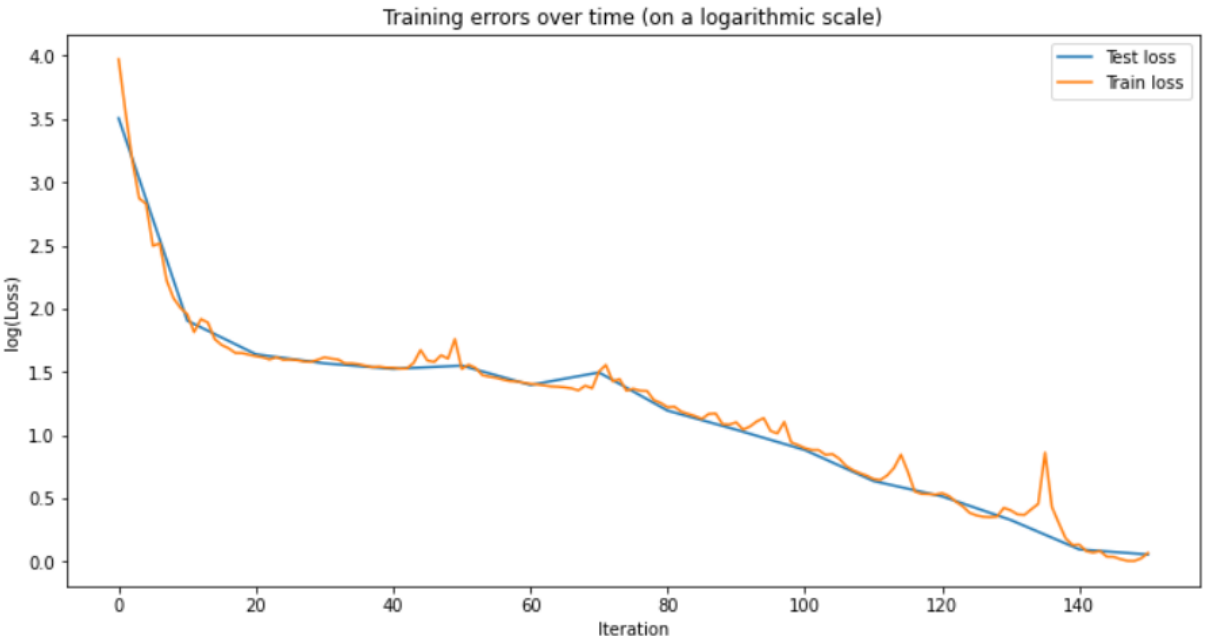
Упражнение 1:

```
# набор обучающих и тестовых данных (sample_x и sample_y соответственно)
# Данные имеют размерность (seq_length , batch_size , output_dim)
sample_x , sample_y = generate_x_y_data_v1(isTrain=True , batch_size=3)

# Длина последовательности (в данных примерах одинаковая для обучающих и тестовых данных)
seq_length = sample_x.shape[0]

# Размер пакета количество(тестовых примеров), по которому усредняется градиент
batch_size = 5
# Размерность выходных данных
output_dim = input_dim = sample_x.shape[-1]
# Количество скрытых нейронов в каждой ячейке
hidden_dim = 12
# Количество ячеек рекуррентной сети (в глубину)
layers_stacked_count = 2
```

```
# Параметры оптимизатора
# Скорость обучения маленькая (скорость обучения позволяет алгоритму не расходиться во время о
бучения)
learning_rate = 0.007
# Количество итераций по обучающей выборке
nb_iters = 150
# Дополнительные параметры алгоритма оптимизации
lr_decay = 0.92
momentum = 0.5
# Коэффициент L2 регуляризации
lambda_l2_reg = 0.003
```

```
Step 0/150, train loss: 53.14033126831055,      TEST loss: 33.3277587890625
Step 10/150, train loss: 7.066632270812988,     TEST loss: 6.71717643737793
Step 20/150, train loss: 5.071545124053955,     TEST loss: 5.149387359619141
Step 30/150, train loss: 5.02582311630249,      TEST loss: 4.792715549468994
Step 40/150, train loss: 4.627418518066406,     TEST loss: 4.586188316345215
Step 50/150, train loss: 4.583797931671143,     TEST loss: 4.708367824554443
Step 60/150, train loss: 4.075402736663818,     TEST loss: 4.03902530670166
Step 70/150, train loss: 4.492512226104736,     TEST loss: 4.459113597869873
Step 80/150, train loss: 3.3865885734558105,    TEST loss: 3.296674966812134
Step 90/150, train loss: 3.005308151245117,     TEST loss: 2.8346266746520996
Step 100/150, train loss: 2.4509406089782715,   TEST loss: 2.4142329692840576
Step 110/150, train loss: 1.9200325012207031,   TEST loss: 1.8892027139663696
Step 120/150, train loss: 1.7167117595672607,   TEST loss: 1.673306941986084
Step 130/150, train loss: 1.4976997375488281,   TEST loss: 1.3880407810211182
Step 140/150, train loss: 1.1420891284942627,   TEST loss: 1.1000385284423828
Step 150/150, train loss: 1.0693156719207764,   TEST loss: 1.056548833847046
Fin. train loss: 1.0693156719207764,    TEST loss: 1.056548833847046
```

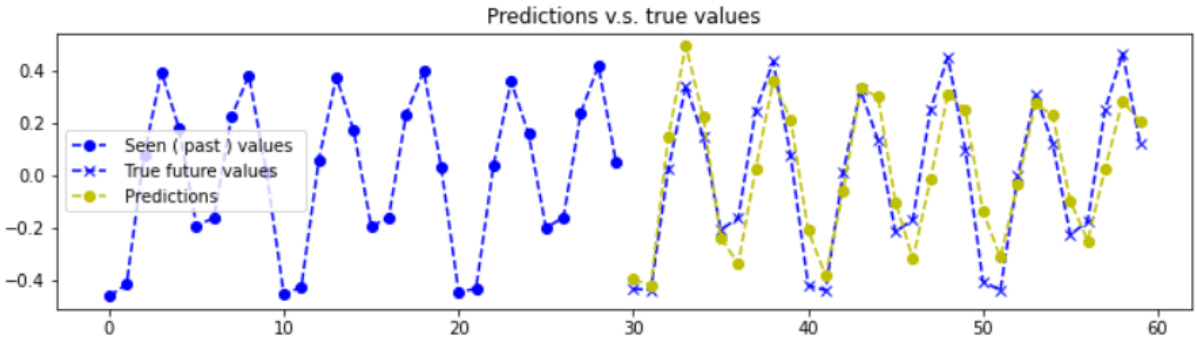




Упражнение 2:

```
# набор обучающих и тестовых данных (sample_x и sample_y соответственно)
# Данные имеют размерность (seq_length , batch_size , output_dim)
sample_x , sample_y = generate_x_y_data_v2(isTrain=True , batch_size=3)

# Длина последовательности (в данных примерах одинаковая для обучающих и тестовых данных)
seq_length = sample_x.shape[0]

# Размер пакета количество(тестовых примеров), по которому усредняется градиент
batch_size = 20
# Размерность выходных данных
output_dim = input_dim = sample_x.shape[-1]
# Количество скрытых нейронов в каждой ячейке
hidden_dim = 60
# Количество ячеек рекуррентной сети (в глубину)
layers_stacked_count = 1

# Параметры оптимизатора
# Скорость обучения маленькая (скорость обучения позволяет алгоритму не расходиться во время о
бучения)
learning_rate = 0.007
# Количество итераций по обучающей выборке
nb_iters = 5000
# Дополнительные параметры алгоритма оптимизации
lr_decay = 0.79
momentum = 0.3
# Коэффициент L2 регуляризации
lambda_l2_reg = 1e-13
```

```
Step 4800/5000, train loss: 91.90878295898438,  TEST loss: 37.290714263916016
Step 4810/5000, train loss: 84.23773193359375,  TEST loss: 52.993202209472656
Step 4820/5000, train loss: 52.42186737060547,  TEST loss: 54.72360610961914
Step 4830/5000, train loss: 50.8092155456543,   TEST loss: 94.63472747802734
Step 4840/5000, train loss: 46.33518981933594,  TEST loss: 51.55702590942383
Step 4850/5000, train loss: 90.78501892089844,  TEST loss: 35.83713150024414
Step 4860/5000, train loss: 39.58888626098633,  TEST loss: 66.44229888916016
Step 4870/5000, train loss: 70.58649444580078,  TEST loss: 76.65387725830078
Step 4880/5000, train loss: 79.96412658691406,  TEST loss: 71.61096954345703
Step 4890/5000, train loss: 66.75050354003906,  TEST loss: 62.48678207397461
Step 4900/5000, train loss: 59.24436569213867,  TEST loss: 68.55169677734375
Step 4910/5000, train loss: 39.801055908203125,         TEST loss: 53.65367889404297
Step 4920/5000, train loss: 55.88743591308594,  TEST loss: 44.61058044433594
Step 4930/5000, train loss: 28.492168426513672,         TEST loss: 53.83274841308594
Step 4940/5000, train loss: 70.76113891601562,  TEST loss: 63.00844955444336
Step 4950/5000, train loss: 53.244468688964844,         TEST loss: 57.329044342041016
Step 4960/5000, train loss: 59.88500213623047,  TEST loss: 72.7364273071289
Step 4970/5000, train loss: 74.57095336914062,  TEST loss: 61.38766860961914
Step 4980/5000, train loss: 37.61663055419922,  TEST loss: 36.62925720214844
Step 4990/5000, train loss: 64.90374755859375,  TEST loss: 69.82931518554688
Step 5000/5000, train loss: 58.52437973022461,  TEST loss: 66.47269439697266
Fin. train loss: 58.52437973022461,     TEST loss: 66.47269439697266
```

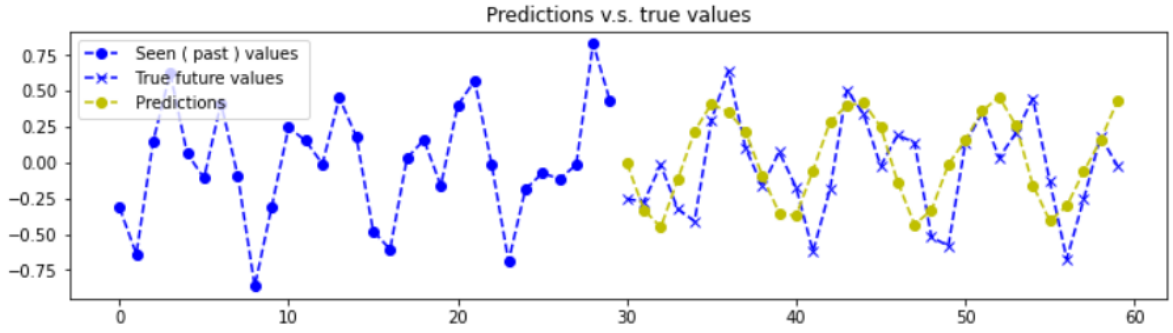


Упражнение 3:

```
# набор обучающих и тестовых данных (sample_x и sample_y соответственно)
# Данные имеют размерность (seq_length , batch_size , output_dim)
sample_x , sample_y = generate_x_y_data_v3(isTrain=True , batch_size=3)

# Длина последовательности (в данных примерах одинаковая для обучающих и тестовых данных)
seq_length = sample_x.shape[0]

# Размер пакета количество(тестовых примеров), по которому усредняется градиент
batch_size = 100
# Размерность выходных данных
output_dim = input_dim = sample_x.shape[-1]
# Количество скрытых нейронов в каждой ячейке
hidden_dim = 50
# Количество ячеек рекуррентной сети (в глубину)
layers_stacked_count = 1

# Параметры оптимизатора
# Скорость обучения маленькая (скорость обучения позволяет алгоритму не расходиться во время о
бучения)
learning_rate = 0.01
# Количество итераций по обучающей выборке
```

```
nb_iters = 2000
# Дополнительные параметры алгоритма оптимизации
lr_decay = 0.91
momentum = 0.3
# Коэффициент L2 регуляризации
lambda_l2_reg = 1e-13
```

```
Step 1900/2000, train loss: 90.9490737915039,   TEST loss: 113.96248626708984
Step 1910/2000, train loss: 85.20500183105469,  TEST loss: 87.00352478027344
Step 1920/2000, train loss: 103.38674926757812,         TEST loss: 126.5726547241211
Step 1930/2000, train loss: 102.91718292236328,         TEST loss: 109.28712463378906
Step 1940/2000, train loss: 101.93095397949219,         TEST loss: 75.9909896850586
Step 1950/2000, train loss: 125.82546997070312,         TEST loss: 90.978271484375
Step 1960/2000, train loss: 61.974510192871094,         TEST loss: 92.66932678222656
Step 1970/2000, train loss: 102.02896118164062,         TEST loss: 89.60110473632812
Step 1980/2000, train loss: 99.11186981201172,  TEST loss: 80.52812194824219
Step 1990/2000, train loss: 97.19345092773438,  TEST loss: 82.3355712890625
Step 2000/2000, train loss: 91.26750946044922,  TEST loss: 105.93649291992188
Fin. train loss: 91.26750946044922,     TEST loss: 105.93649291992188
```



Выводы

В ходе лабораторной работы были получены навыки работы с рекуррентными нейронными сетями в tensorflow.

Контрольные вопросы

1. В чем преимущество рекуррентных нейронных сетей по сравнению с обычными персептронами?
   В рекуррентных нейронных сетях связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети RNN применимы в таких задачах, где нечто целостное разбито на части, например: распознавание рукописного текста или распознавание речи.

2. Что такое регуляризация и зачем она нужна?

   Регуляризация используется для обозначения ограничения оптимизации путем наложения штрафа на сложность решения в попытке предотвратить переобучение на обучающей выборке.

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

   Пакетный градиентный спуск – алгоритм градиентного спуска, который позволяет вычислить ошибку для каждого примера в обучающем наборе данных, но обновляет модель только тогда, когда все обучающие примеры оценены.

   Мини-пакетный градиентный спуск – алгоритм градиентного спуска, разбивающий обучающий набор данных на небольшие партии, использующиеся для расчета ошибки модели и обновления её коэффициентов.

   Онлайновый градиентный спуск – алгоритм градиентного спуска, при котором обновление параметров каждой точки модели происходит после оценки каждого примера обучающего набора данных.

Список литературы

[1] tensorfow. https://www.tensorflow.org/.

[2] J. Brownlee. Encoder-decoder recurrent neural network models for neural machine translation.

https://machinelearningmastery.com/ encoder-decoder-recurrent-neural-network-models-neuralmachine-translation/.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016.

[4] Архангельская Е. Николенко С., Кадурин А. Глубокое обучение. Погружение в мир

нейронных сетей. СПб.: Питер, 2018.