Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Отчет по лабораторной работе №8

по курсу «Нейроинформатика»

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М8О-407Б, № по списку 20.

Преподаватель: Аносова Н.П.

Дата: 27.12.2021

Итоговая оценка:

Подпись преподавателя:

Москва, 2021

Тема лабораторной: “Динамические сети”.

Целью работы является исследование свойств некоторых динамических нейронных сетей, алгоритмов обучения, а также применение сетей в задачах аппроксимации функций и распозна- вания динамических образов.

Основные этапы работы:

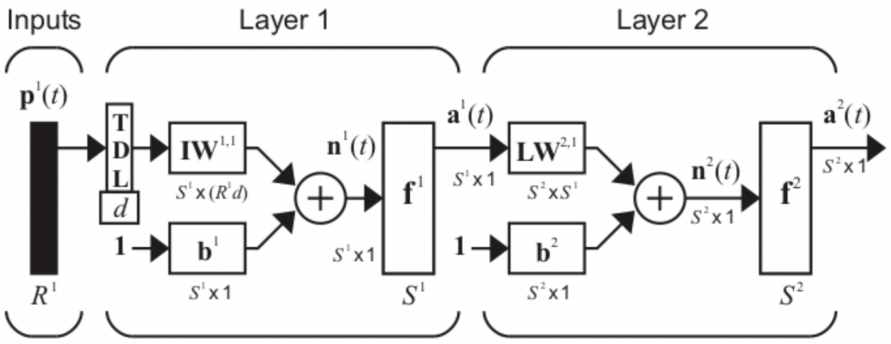
1. Использовать сеть прямого распространения с запаздыванием для предсказания значений временного ряда и выполнения многошагового прогноза.
2. Использовать сеть сеть прямого распространения с распределенным запаздыванием для распознавания динамических образов.
3. Использовать нелинейную авторегрессионную сеть с внешними входами для аппроксимации траектории динамической системы и выполнения многошагового прогноза.

20.	05/1874

20.	$u(k) = \sin(2k^2 - 6k - \pi)$

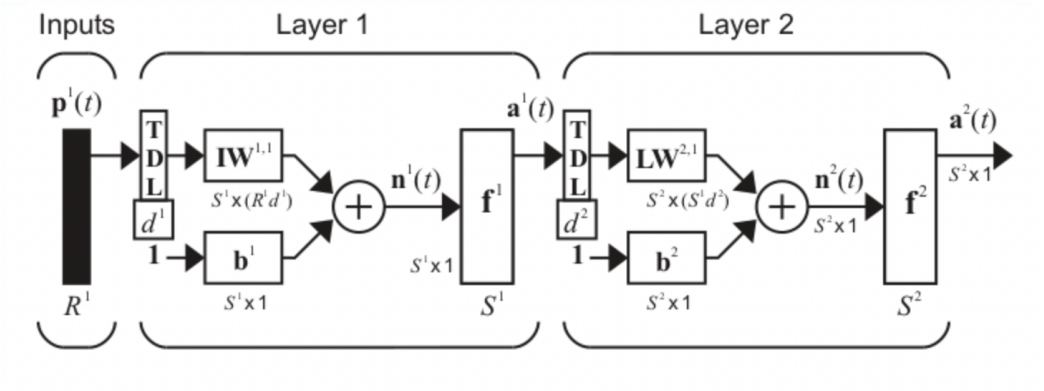
Ход работы

Сеть прямого распространения с запаздыванием похожа на адаптивный фильтратор, однако имеет 2 полносвязных слоя вместо одного, что делает ее более гибкой.

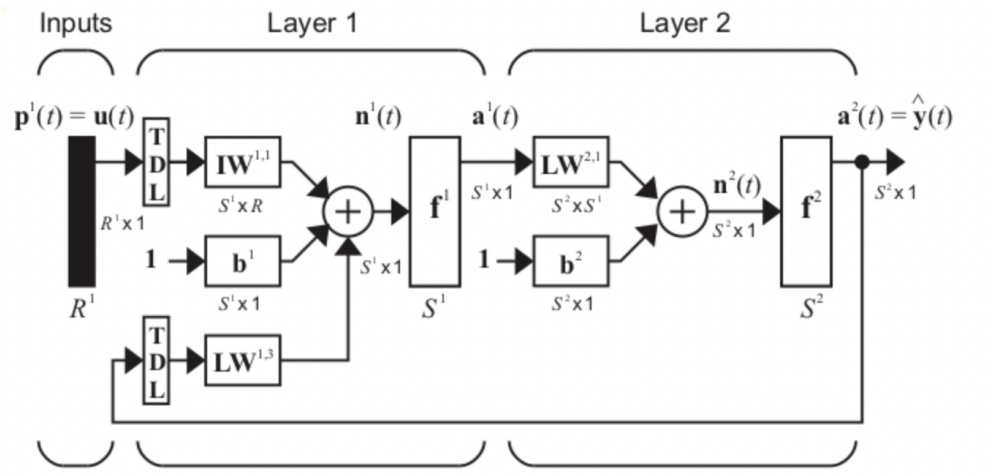


Такая система позволяет успешно справляться с задачами, связанными с динамическими процессами, такими как распознавание звукового или видео потока.
TDL - Tapped Delay Line. Модуль линий задержек.

Сеть прямого распространения с распределенным запаздыванием имеет *TDL* блок не только перед первым слоем, но и перед вторым.



Нелинейная авторегрессионная сеть с внешними входами отличается тем, что выход первого слоя формируется не только из перемножения матрицы весов на входные значения, но и из перемножения другой матрицы весов с выходом *TDL* блока, сформированного из предыдущих выходов нейронной сети.

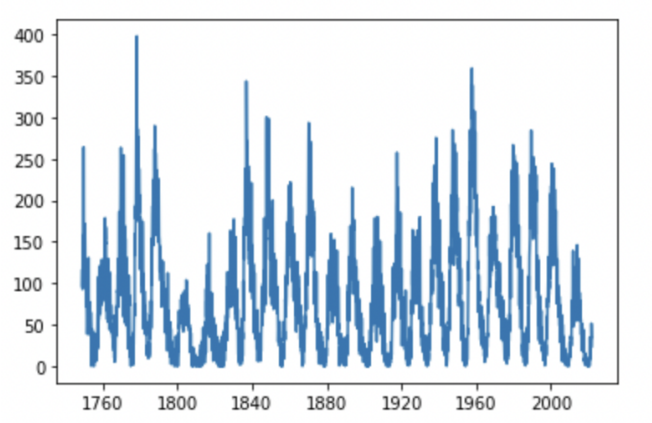


Помогает в адаптации.

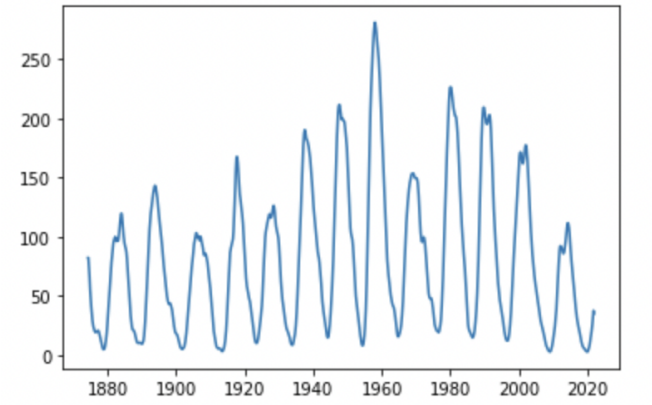
Ход работы

Задание 1

Числа Вольфа до преобразований:



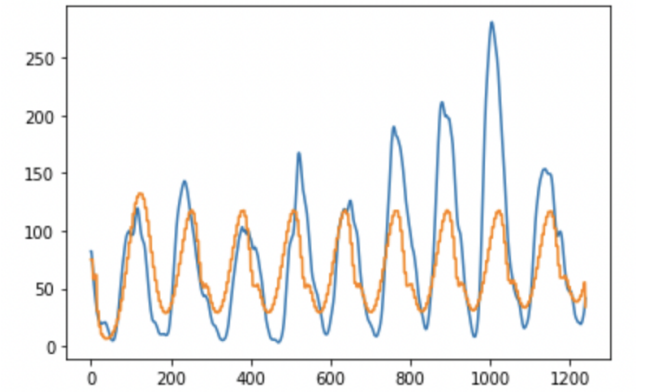
Числа Вольфа после сглаживания и среза от требуемой даты:



Результаты обучения. Обучающая выборка:

```
output = pyrenn.NNOut(np.array(X_t), nn)
MSE = mean_squared_error(output, y_train)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))
```

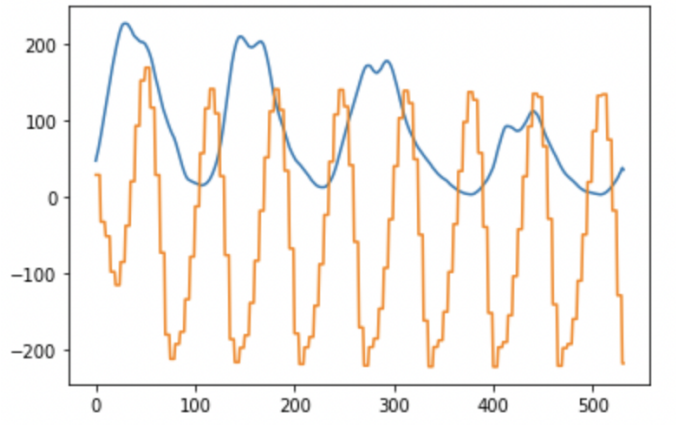
```
MSE = 2053.8661761905146
RMSE = 45.319600353384786
```



Тестовая выборка:

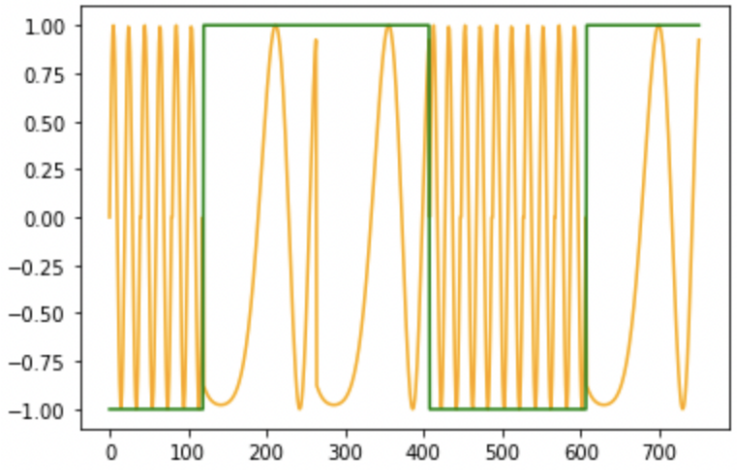
```
output = pyrenn.NNOut(np.array(X_t), nn)
MSE = mean_squared_error(output, y_test)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))
```

```
MSE = 36431.80089910292
RMSE = 190.87116308940676
```

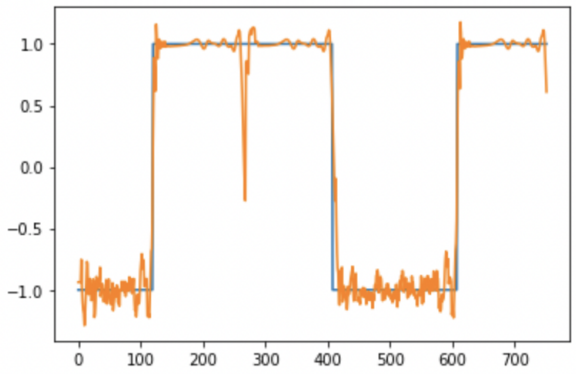


Задача 2

Входное множество.



Результаты на обучающей выборке:

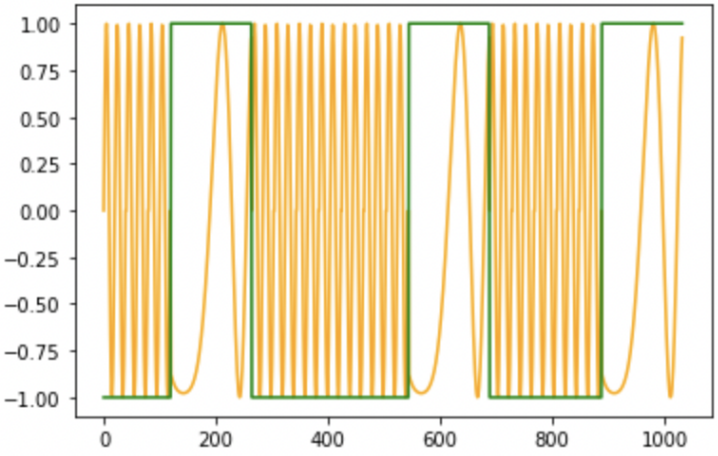


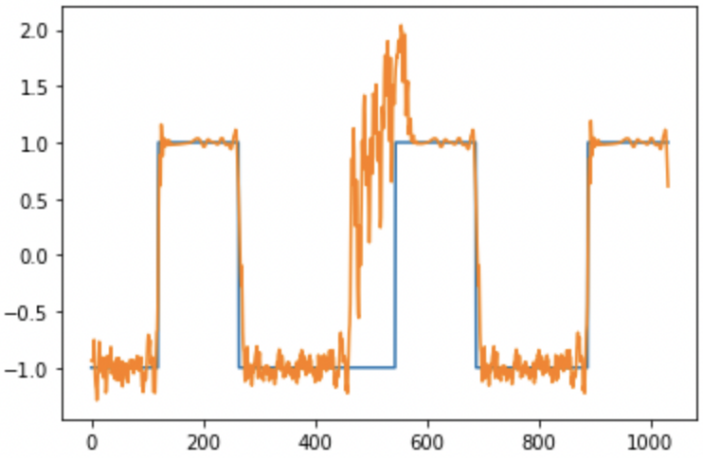

```
output[output >= 0] = 1.0
output[output < 0] = -1.0

MSE = mean_squared_error(T.reshape(T.shape[0]), output)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))
```

```
MSE = 0.026595744680851064
RMSE = 0.16308201826336055
```

Изменив второй элемент r на 7:





Задание 3

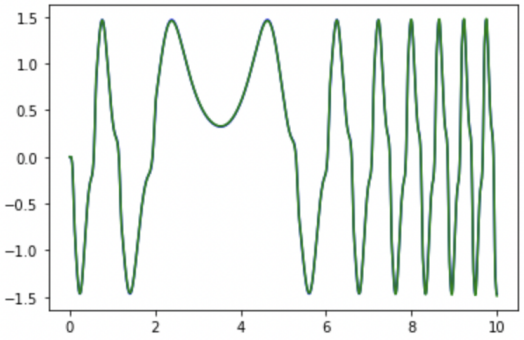
```
output = narx.predict(inpt.reshape(-1, 1), target, step=1)

output[np.isnan(output)] = 0
MSE = mean_squared_error(target, output)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))
```

```
MSE = 0.00018018168863415308
RMSE = 0.01342317729280788
```

```
plt.plot(k, y, color='blue')
plt.plot(k, output, color='green')
```

```
[<matplotlib.lines.Line2D at 0x7f9f92164910>]
```



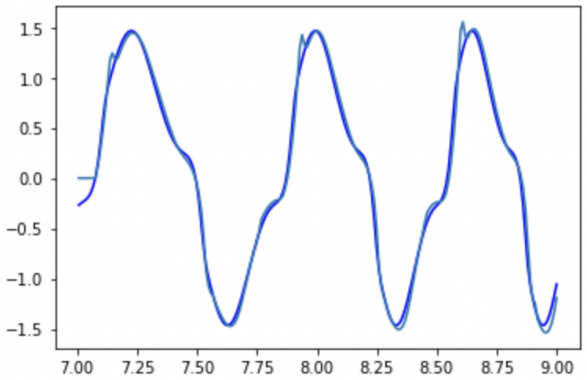
Многошаговый прогноз:

```
outputTest[np.isnan(outputTest)] = 0
MSE = mean_squared_error(targetTest, outputTest)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))
```

```
MSE = 0.008025981352681868
RMSE = 0.0895878415449433
```

```
plt.plot(xTest, yTest, color='blue')
plt.plot(xTest, outputTest)
```

```
[<matplotlib.lines.Line2D at 0x7f9f920f8050>]
```



Выводы

Для использования NARX пришлось залезать в исходники и копировать классы и функции, потому что установить работающую версию через pip не удалось.

Статические НС-модели *VS* Динамические НС-модели

Статические сети: отсутствуют обратные связи и задержки, выход таких сетей вычисляется непосредственно по их входам с помощью вычислений «распространяющихся вперед», от входов к выходам.

Динамические сети: выход зависит не только от текущего входа сети, но и от ее предшествующих (по времени) входов, выходов и/или состояний.

Динамические сети:

- рекуррентные сети с обратными связями в них;
- сети прямого распространения с линиями задержки в них;
- рекуррентные сети с обратными связями и линиями задержки.