САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет
Кафедра информатики

Лысов Александр Васильевич
Епрев Артем Евгеньевич

Курсовая работа

Применение LSTM RNN для предсказания состояния литий-ионных батарей

3 курс, группа 342

Руководитель курсовой работы
_____ А. А. Алиев
«____» _____ 2017 г.

Санкт-Петербург, 2017 г.

Содержание

1 Введение

Целью нашей работы являлось исследование работы литий-ионных батарей.

Литий-ионные батареи используется повсюду: от наручных часов до электрических автомобилей, поэтому исследования в данной области имеют большой потенциал для применения в реальном мире.

В данной курсовой работе будет исследована зависимость емкости батареи от ее поведения во время разряда, а также построена модель учитывающая эту зависимость.

2 Основная часть

2.1 Начальный набор данных

Были взяты наборы данных циклов зарядов-разрядов батарей. Набор данных был взят с сайта ti.arc.nasa.gov[1] и был представлен в расширении .mat.

Набор из четырех литий-ионных батарей (№ 5, 6, 7 и 18) выполнялся через 2 различных рабочих профиля (заряд и разряд) при комнатной температуре. Зарядка проводилась в режиме постоянного тока (CC) при 1,5 A до тех пор, пока напряжение батареи не достигло 4,2 В, а затем продолжалось в режиме постоянного напряжения (CV), пока зарядный ток не упал до 20 мА.

Разряд проводился при постоянном токе (CC), равном 2A, до тех пор, пока напряжение батареи не упало до 2,7 В, 2,5 В, 2,2 В и 2,5 В для батарей 5 6 7 и 18 соответственно.

Эксперименты были прекращены, когда батареи достигли критериев окончания срока службы (EOL), то есть на 30% снизились в номинальной емкости (от 2Ahr до 1.4Ahr).
В датасете были следующие параметры для соответствующих циклов:

charge:

Voltage_measured	Battery terminal voltage (Volts)
Current_measured	Battery output current (Amps)
Temperature_measured	Battery temperature (degree C)
Current_charge	Current measured at charger (Amps)
Voltage_charge	Voltage measured at charger (Volts)
Time	Time vector for the cycle (secs)

discharge:

Voltage_measured	Battery terminal voltage (Volts)
Current_measured	Battery output current (Amps)
Temperature_measured	Battery temperature (degree C)
Current_charge	Current measured at load (Amps)
Voltage_charge	Voltage measured at load (Volts)
Time	Time vector for the cycle (secs)
Capacity	Battery capacity (Ahr) for discharge till 2.5V

[1] https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/#battery

2.2 Обработка данных

После загрузки набора данных в расширении .mat была получена неудобная для дальнейшего использования структура. Было решено переформатировать ее в расширение .csv со структурой следующего вида:

x_dataset					y_dataset
label1_1;	label1_2;	label1_3;	⋯;	label1_N	res1
label2_1;	label2_2;	label2_3;	⋯;	label2_N	res2
⋮			⋱		⋮
labelM_1;	labelM_2;	labelM_3;	⋯;	labelM_N	res3

и

Реализация тренировочного и тестового датасетов входных данных:

```
fileName="B0006"
pathDS="ds/"
a = loadmat("d1/"+fileName+".mat")
newDS=open(pathDS+"xd_train.csv", "w")
for i in range(616):
    numbOfFeat=len(a[fileName][0, 0][0][0][i][3][0][0])
    numbOfVect=len(a[fileName][0, 0][0][0][i][3][0][0][0][0])
    typeCycle=a[fileName][0, 0][0][0][i][0][0]
    if (not typeCycle=="discharge"):
        continue
    ar=np.linspace(0, numbOfVect-1, 10, dtype=int)
    for j in ar:
        for k in range (6):
            newDS.write(str(a[fileName][0, 0][0][0][i][3][0][0][k][0][j]))
            if (not (j==ar[-1] and k==5)):
                newDS.write(";")
    newDS.write("\n")
newDS.close()
```

Реализация тренировочного и тестового датасетов целевых переменных:

```
pathDS="ds/"
fileName="B0006"
xS=1
a = loadmat("d1/"+fileName+".mat")
newDS=open(pathDS+"y_train.csv", "w")
for i in range(616):
    numbOfFeat=len(a[fileName][0, 0][0][0][i][3][0][0])
    typeCycle=a[fileName][0, 0][0][0][i][0][0]
    if (not typeCycle=="discharge"):
        continue
    for h in range(xS):
        newDS.write(str(a[fileName][0, 0][0][0][i][3][0][0][6][0][0])+"\n")
newDS.close()
```

2.3 Идеи методов

Предварительно исследовав изменения значения параметра емкости батареи на протяжении всех циклов (изменение было как минимум не линейным)1, а также учитывая большое количество признаков в каждом из объектов, было решено использовать нейронные сети.

Построение нейронных сетей явлется одним из самых передовых и хорошо зарекомендовавших себя подходов в машинном обучении. Так как для адекватного предсказания требовалось учитывать структуру данных (временной ряд), было решено использовать рекуррентную нейронную сеть с долгой краткосрочной памятью (recurrent neural network long short-term memory)2.
Наш метод решения данной проблемы состоит из нескольких этапов:

1. Поиск наиболее подходящего набора данных (датасета);
2. Обработка датасета, перевод из структуры MatLab (.mat) в привычный .csv;
3. Построение графиков зависимости емкости батареи от параметров циклов разряда;
4. Перевод каждого цикла разряда, в каждом из которых содержалось различное количество векторов состояний, в один вектор, понятный для алгоритма;
5. Построение LTSM рекуррентной нейросети;
6. Отбор наилучших параметров алгоритма;
7. Проверка точности на тестовых данных.

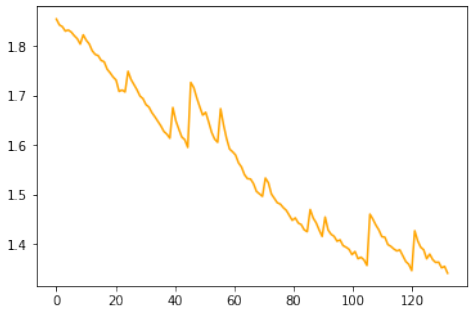

Рис. 1: График зависимости capacity от номера цикла разряда

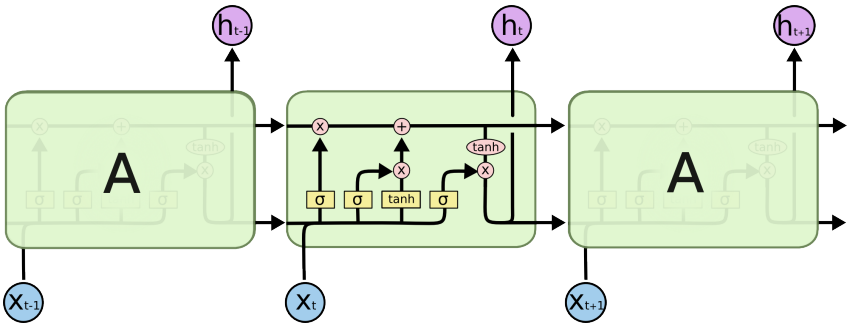

Рис. 2: Принцип LSTM рекуррентной нейронной сети

2.4 Реализация метода

Программа написана на языке Python2, и в ней использовались библиотеки keras, numpy, matplotlib, pandas, sklearn и scipy.

Чуть подробнее об использовании каждой из библиотек:

Keras Библиотека глубинного обучения с достаточным количеством метрик, оптимизаторов и функций потерь, которая может применяться на основе Theano или TensorFlow. Было решено использовать backend Theano, чтобы попробовать распараллелить на CUDA, но, к сожалению, имеющаяся видеокарта NVidia GeForce 610M оказалась слишком старой и значительного прироста это не дало, только ускорение в 1.5—2 раза;

NumPy Библиотека, из которой были использованы numpy-массивы для ускорения работы и упрощения реализации программы;

Matplotlib Библиотека для построения графиков;

Pandas Была использована для загрузки .csv датасетов;

Sklearn Еще одна библиотека, используемая для машинного обучения. Для качества измерения использовались различные метрики из нее;

SciPy Была нужна для загрузки изначального .mat набора данных.

Процесс реализации состоял из следующих этапов:

1. Загрузка набора данных:
   При помощи метода loadmat из scipy.io был загружен изначальный набор данных циклов батарей. Структура была неудобна для дальнейшего использования, а именно, имела вид:

   ```
   dataset[fileName][0, 0][0][0][i][3][0][0][k][0][j]
   ```

   где i — номер цикла, k — номер вектора в цикле, j — номер параметра в векторе.

2. Выбор тренировочного и тестового датасетов:
   Так как батареи 6 и 18 разряжались до одинакового напряжения (2.5 В), было решено выбрать батарею 6 в качестве тренировочного датасета, а 18 в качестве тестового датасета.

   - Создание тренировочного и тестового датасетов объектов:
     Поскольку один цикл разряда представлял собой матрицу Nx6, где N был от 180 до 371, было решено брать 10 равнораспределенных векторов при помощи:

     ```
     np.linspace(0, numbOfVect-1, 10, dtype=int)
     ```

     и объединять их в один для каждого цикла. В итоге получилось два набора данных: x_train (168x60) и x_test (132x60);
   - Создание тренировочного и тестового датасетов целевой переменной:
     В каждом цикле разряда также содержалось одно значение емкости батареи (capacity). Целью построенной рекуррентной нейронной сети являлось предсказание данного значения по входному вектору состояния цикла разряда батареи.
     Были также созданы 2 набора данных целевой переменной: y_train (168x1) и y_test (132x1);.

3. Построение модели:
   В ходе построения LSTM рекуррентной нейросети были использованы слои:

   keras.layers.Dense(60, input_shape=(60,))

   keras.layers.Reshape((60,1)))

   keras.layers.LSTM(60, return_sequences=True)

   keras.layers.LSTM(60, dropout=0.31, recurrent_dropout=0.32)

   keras.layers.Dense(1)

   В качестве функции потерь (loss) была выбрана среднеквадратическая ошибка, в качестве оптимизатора (optimizer) был выбран adam, поскольку они оказались наиболее подходящими для решения поставленной задачи.

4. Выбор оптимальных параметров:
   Помимо выбора непосредственно структуры сети, требовалось подобрать оптимальные параметры такие как:

   - Epochs[2]. В качестве оптимального было выбрано значение 21.
   - Dropout[3] В качестве оптимального было выбрано значение 0.31.
   - В качестве recurrent_dropout было выбрано 0.32.

5. Оценка точности реализованной модели:
   После применения реализованной модели к тестовым данным значение функции потерь mean_square_error составило 0.000784.

[2]Epochs — количество итераций в процессе обучения. Влияют на степень обученности модели. Большое количество эпох может привести к переобучению на тренировочных данных и, соответственно, плохим результатам на тестовой выборке.

[3]Dropout — параметр отвечающий за переобучение. Он обнуляет случайную заданную долю признаков и мешает коадаптации весов в слоях.

3 Заключение

В итоге обученная рекуррентная нейронная сеть предсказала результаты3 со среднеквадратической ошибкой на тестовых данных, равной 0.000784

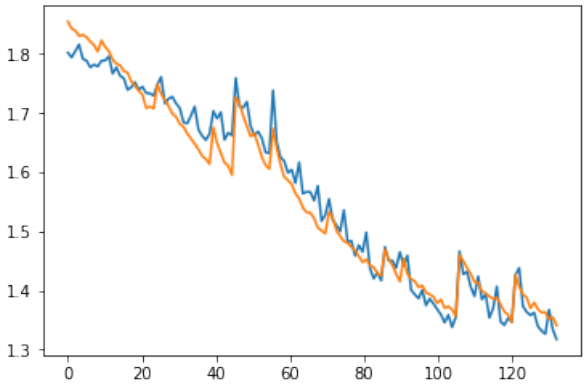

Рис. 3: Сравнение значений. Желтым — тестовые данные, синим — результат предсказания реализованной рекуррентной нейросети

Несмотря на то что тренировочных данных было мало, обученная реализация модели оказалась способной в достаточной степени предсказывать результат.

В дальнейшем, это исследование может помочь предсказывать оставшееся время жизни батареи по ее состоянию во время цикла разряда.

Исходный код курсовой работы находится здесь: https://github.com/lysa0/coursework

Список литературы

1. https://en.wikipedia.org/wiki/Long_short-term_memory
2. http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
3. https://keras.io/metrics/
4. https://keras.io/losses/
5. https://keras.io/optimizers/
6. https://keras.io/models/model/
7. https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/#battery