FAKULTETA ZA INFORMACIJSKE ŠTUDIJE V NOVEM MESTU

Superračunalnik za vse

–

Izračun linearne regresije na superračunalniku Rudolf z uporabo TensorFlow orodja in podatkov o čistosti zraka

PO KREATIVNI POTI

PROJEKT: SUPERRAČUNALNIK ZA VSE

PRIPRAVIL: Dejan Penko

MENTOR: doc. dr. Biljana Mileva Boshkoska, doc. dr. Leon Kos

Ljubljana, julij 2017

POVZETEK

V tem poročilu je predstavljen postopek izvedbe izračuna linearne regresije na superračunalniku Rudolf na Fakulteti za Informacijske Študije v Novem Mestu z uporabo odprtokodnega orodja TensorFlow orodja ter podatkov o čistosti zraka. Predstavljene so tudi potrebne programske kode za pridobitev podatkov v ustrezni obliki, potek same linearne regresije ter grafični prikaz rezultatov.

KLJUČNE BESEDE: PKP projekt Superračunalnik za vse, podatki o čistost zraka, linearna regresija, Linux, Python, TensorFlow

KAZALO

KAZALO SLIK IN TABEL

1 UVOD

Namen tega poročila je predstaviti reševanje konkretne naloge, v tem primeru izračun linearne regresije čistosti zraka z uporabo orodja TensorFlow na superračunalniku Rudolf, kar vključuje tudi krajšo obrazložitev več programskih kod potrebnih za pripravo podatkov, njihovo obdelavo oz. izračun na superračunalniku Rudolf ter za prikaz rezultatov.
Za testne podatke se je uporabilo podatke o čistosti zraka, kateri so na voljo na spletni strani http://www.arso.gov.si/en/air/data/, natančneje za dnevno koncentracijo delcev PM2.5 (oz. delcev večjih od 2.5 μg/m3) v zraku skozi leto 2016, na voljo na povezavi http://www.arso.gov.si/en/air/data/PM2.5_D_dec16_ang.pdf .

2 PRIPRAVA PODATKOV O ČISTOSTI ZRAKA

Uporabljeni podatki so shranjeni v obliki tabele v .pdf datoteki, prikazano v Slika 1, naša programska koda za izračun linearne regresije, predstavljena v naslednjem poglavju, pa zahteva podatke zapisane v .txt datoteki v obliki, kot je prikazano v Slika 2.

Date/ Measuring site	Ljubljana BF	Maribor center	MB Vrbanski plato	Iskrba
01.01.16	59	81	72	18
02.01.16	75	66	63	28
03.01.16	40	40	38	30
04.01.16	49	54	52	30
05.01.16	93	96	92	27
06.01.16	67	74	76	20
07.01.16	37	50	40	7
08.01.16	64	92	78	8
09.01.16	82	52	51	3
10.01.16	58	44	43	5

Slika 1: Oblika zapisa podatkov v .pdf datoteki

```
##, PM2.5
##, Date/,,,,
##, "",Ljubljana BF,Maribor center,MB Vrbanski
plato,Iskrba
1, 01.01.16,59,81,72,18
2, 02.01.16,75,66,63,28
3, 03.01.16,40,40,38,30
4, 04.01.16,49,54,52,30
5, 05.01.16,93,96,92,27
6, 06.01.16,67,74,76,20
7, 07.01.16,37,50,40,7
8, 08.01.16,64,92,78,8
9, 09.01.16,82,52,51,3
10, 10.01.16,58,44,43,5
```

Slika 2: Potrebna oblika podatkov v .txt datoteki

Lastnosti .txt datoteke, katera vsebuje podatke o čistosti zraka:

- »**##,**« označuje vrstice z dodatnimi podatki, kateri niso vključeni v sam izračun linearne regresije
- Prvi vnos v vrstici: številka dneva v letu
- Drugi vnos v vrstici: datum zapisan v obliki dd.mm.yy
- Tretji vnos v vrstici: številčni podatek za prvi kraj

- Četrti vnos v vrstici: številčni podatek za drugi kraj itd.

Ročna pretvorba podatkov iz .pdf datoteke v .txt datoteko v zgoraj predstavljeno obliko je izvedljiva, vendar zamudna in težavna, zato se je za pretvorbo podatkov ustvarilo program, imenovan pdfTable2txt in zapisan v programskem jeziku Python 2.7, kateri to delo opravi namesto nas.

Kratek opis uporabe in poteka programa **pdfTable2txt.py**:

- Predno lahko uporabimo program je potrebno namestiti Python paket imen. **Tabula.** To storimo z naslednjimi ukazi:

```
$ sudo apt-get install python-pip
$ pip install tabula-py
```

- Program poženemo v Linux terminalu z ukazom

```
$ python pdfTable2txt.py <datoteka.pdf>
```

  kjer **<datoteka.pdf>** predstavlja ime .pdf datoteke s tabelo, katero želimo pretvoriti v .txt datoteko.
- Funkcija **pdf2csv()** prebere .pdf datoteko in jo pretvori v .csv datoteko s pomočjo python paketa imen. »tabula«.
- Funkcija **checkWhichParticles()** ugotovi na katere delce se navezujejo podatki. Ta podatek se potrebuje pozneje.
- Funkcija **csv2txt()** pretvori prej ustvarjeno .csv datoteko v končno .txt datoteko, katera vsebuje podatke v željeni obliki.

Program **pdfTable2txt.py**:

```
1   # Author: Dejan penko
2   # Linux terminal run command:
3   # $ python pdfTable2txt.py <input_file.pdf>
4   # This script reads table .pdf file and converts the table to
5   # .txt file named <input_file.txt>.
```

```
6
7   import tabula
8   import sys
9   import os.path
10  import PyPDF2
11
12  def pdf2csv(pdfname, csvname):
13      filename_pdf = str(pdfname)
14      filename_csv = str(csvname)
15      # Convert whole (all pages) pdf table
16      tabula.convert_into(filename_pdf, filename_csv,  \
17          output_format="csv", pages="all")
18
19  def csv2txt(csvname, txtname, particles_label):
20      # Convert .csv to adjusted .txt file
21      filename_csv = str(csvname)
22      filename_txt = str(txtname)
23      f = open(filename_csv, "r")
24      lines = f.readlines()
25      f.close
26      f = open(filename_txt, "w")
27      d = 1
28      line_count = 0
29      f.write('##, '+particles_label+'\n')
30      for line in lines:
31          # Include only few first header lines, skip others.
32          # Then write only data fields (recognized by the
33          # first character in line being a number.
34          line_count+=1
35          first_char_is_digit = line[0].isdigit()
36          if first_char_is_digit==False:
37              if line_count < 3:
38                  f.write('##, ' + line)
39              else:
40                  continue
41              continue
42          # Write the "cleaned" .txt file, with
43          # additional day of the year (1, 2, 3 ... n_days) in front of the line
44          f.write(str(d) + ', ' + line)
45          d+=1
46      f.close()
47
```

```
48  def checkWhichParticles(filename):
49      pdfFileObj = open(filename, 'rb')
50      pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
51      pageObj = pdfReader.getPage(0)
52      all_text = pageObj.extractText()
53      all_text=''.join(all_text.split())
54      if "PM2.5" in all_text:
55          particles_label = "PM2.5"
56      elif "PM10" in all_text:
57          particles_label = "PM10"
58      return particles_label
59
60  def main(argv):
61      if not argv:
62          print "Enter input pdf filename and output csv and txt filename."
63          sys.exit()
64
65      pdfname = argv[1]
66      csvname = os.path.splitext(pdfname)[0] + '.csv'
67      txtname = os.path.splitext(pdfname)[0] + '.txt'
68      pdf2csv(pdfname, csvname)
69      particles_label=checkWhichParticles(pdfname)
70      csv2txt(csvname, txtname, particles_label)
71
72  if __name__ == '__main__':
73      main(sys.argv)
```

3 POTEK LINEARNE REGRESIJE

Program za izračun linearne regresije iz pripravljenih podatkov, prav tako zapisan v Python2.7 programskem jeziku, se imenuje **slr.py**. Ta program vsebuje posodobljeno ter nadgrajeno programsko kode dosegljivo na spletni strani https://github.com/e-lin/tensorflow_practices/blob/master/examples/linear_regression.py , katera uporablja Python paket TensorFlow, odprtokodnega programskega orodja za numerične izračune z uporabo podatkov iz grafov podatkovnega poteka.

Kratek opis uporabe in poteka programa **slr.py**:

- Program poženemo v Linux terminalu z ukazom

```
$ python slr.py -i <inputfile> -c <column> -m <month_ang>
```

kjer **<inputfile>** predstavlja ime .txt datoteke s podatki, **<column>** prestavlja stolpec s podatki oz. kraj, katerega podatke želimo uporabiti ter **<month_ang>** predstavlja ime meseca, zapisanega v angleščini in z malimi tiskanimi črkami (npr. january).

- Funkcija **month2number()** ustrezno pretvori ime meseca v ustrezno obliko za primerjavo s podatki iz .txt datoteke (npr. »januar« pretvori v »01«)
- Funkcija **read_data()** prebere .txt datoteko, izlušči željene podatke in jih shrani v ustrezno obliko
- V funkciji **run_slr()** poteče izračun linearne regresije z uporabo izluščenih podatkov ter uporabo TensorFlow python paketa. Rezultati so nato shranjeni v ločene datoteke imen. **o_data_X.txt**, **o_data_Y.txt** in **o_fLine.txt**.

Program **slr.py**:

```
1  # Author: Dejan Penko
2  # Linear regression with one variable by using Tensorflow.
3  # Python 2.7
4  # Example run command:
5  # python slr.py -i <data.txt> -c <column> -m <month>
6  # where column is column number and month is month name
7  # written in lowercase (jan or januar etc.)
8
9  import numpy as np
10 import tensorflow as tf
11 import datetime
12
13 # model parameters as external flags
14 flags = tf.app.flags
15 FLAGS = flags.FLAGS
16 flags.DEFINE_float('learning_rate', 0.01, 'Initial learning rate.')
17 flags.DEFINE_integer('max_steps', 2000, 'Number of steps to run trainer.'
18 flags.DEFINE_integer('display_step', 50, 'Display logs per step.'
19
20
21 X = tf.placeholder(tf.float32)
22 Y = tf.placeholder(tf.float32)
23
```

```
24  def run_slr(data_X, data_Y):
25      m = data_X.shape[0] # number of examples
26
27      # weights
28      W = tf.Variable(0.0, name="weight")    # 0.0 by default
29      b = tf.Variable(0.0, name="bias")      # 0.0 by default
30
31      # linear model
32      activation = tf.add(tf.multiply(X, W), b)
33      calc = tf.reduce_sum(tf.square(activation - Y)) / (2*m)
34      optimizer = tf.train.GradientDescentOptimizer(FLAGS.learning_rate).minimize(calc)
35
36      with tf.Session() as sess:
37          init = tf.initialize_all_variables()
38          sess.run(init)
39          for step in range(FLAGS.max_steps):
40              for (x, y) in zip(data_X, data_Y):
41                  sess.run(optimizer, feed_dict={X: x, Y: y})
42                  if step % FLAGS.display_step == 0:
43                    print "Step:", "%04d" % (step+1), "Calc=", "{:.9f}".format(sess.run(calc, \
44                        feed_dict={X: data_X, Y:data_Y})), "W=", sess.run(W), "b=", sess.run(b)
45
46          print "Optimization Finished!"
47          data_calc = sess.run(calc, feed_dict={X: data_X, Y: data_Y})
48          print "Calculation=", data_calc, "W=", sess.run(W), "b=", sess.run(b), '\n'
49          # Save to multiple .txt file
50          np.savetxt("o_data_X.txt", data_X, fmt='%2.8f')
51          np.savetxt("o_data_Y.txt", data_Y, fmt='%2.8f')
52
53          fLine = np.empty(m) * np.nan
54          fLine = sess.run(W) * data_X + sess.run(b)
55          np.savetxt("o_fLine.txt", fLine, fmt='%2.8f')
56
57  def month2number(month_name):
58      if "jan" in month_name:
59          return "01"
60      elif  "feb" in month_name:
61          return "02"
62      elif  "mar" in month_name:
63          return "03"
64      elif  "apr" in month_name:
65          return "04"
```

```
66      elif  "may" in month_name:
67          return "05"
68      elif  "jun" in month_name:
69          return "06"
70      elif  "jul" in month_name:
71          return "07"
72      elif  "aug" in month_name:
73          return "08"
74      elif  "sep" in month_name:
75          return "09"
76      elif  "oct" in month_name:
77          return "10"
78      elif  "nov" in month_name:
79          return "11"
80      elif  "dec" in month_name:
81          return "12"
82      else:
83          return "all"
84
85  def read_data(filename, c, month="all", read_from_file = True):
86      # Convert month name into month number
87      if month != "all":
88          month = month2number(month)
89      # Read data from input .txt file
90      if read_from_file:
91          with open(filename) as fd:
92              data_list = fd.read().splitlines()
93              m = len(data_list) # number of examples
94              data_X = np.array([])
95              data_Y = np.array([])
96              for i in range(m):
97                  line_element = data_list[i].split(",")
98                  data_element = line_element[1].split(".")
99
100                 # In some cases the month in data format dd.mm.yy is in
101                 # input.txt file  not written correctly,
102                 # for example "5" instead "05".
103                 # In that case there is only one character, and to make the code
104                 # work with no errors we add
105                 # the missing "0" in the date number
106                 if len(data_element[1]) == 1:
107                     month = "0"+data_element[1]
```

```
                if data_element[1] == str(month):
                    if (line_element[int(c)] == '-') or (line_element[0] ==##'):
                        print "skipping line: ", i
                        continue
                    else:
                        data_X = np.append(data_X, float(line_element[0]))
                        data_Y = np.append(data_Y, [float(line_element[int(c)])])
                # If month was not specified, take all available data
                elif month == "all":
                    if (line_element[int(c)] =='-') or (line_element[0] =='##'):
                        print "skipping line: ", i
                        continue
                    else:
                        data_X = np.append(data_X, float(line_element[0]))
                        data_Y = np.append(data_Y, [float(line_element[int(c)])])
    return data_X, data_Y

import sys, getopt

def main(argv):
    try:
        opts, args = getopt.getopt(argv,"hi:c:m:",["ifile=","clm=", "month"])
    except getopt.GetoptError:
        print 'Run command: python slr.py -i <inputfile> -c <column> -m <month_ang>'
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            Print 'Run command: python slr.py -i <inputfile> -c <column> -m <month_ang>'
            sys.exit()
        elif opt in ("-i", "--ifile"):
            inputfile = arg
        elif opt in ("-c", "--clm"):
            column_of_values = arg
        elif opt in ("-m", "--month"):
            month = arg
    data_X, data_Y = read_data(inputfile, column_of_values, month)
    run_slr(data_X, data_Y)

if __name__ == '__main__':
    #tf.app.run()
    main(sys.argv[1:])
```

4 GRAFIČNI PRIKAZ REZULTATOV

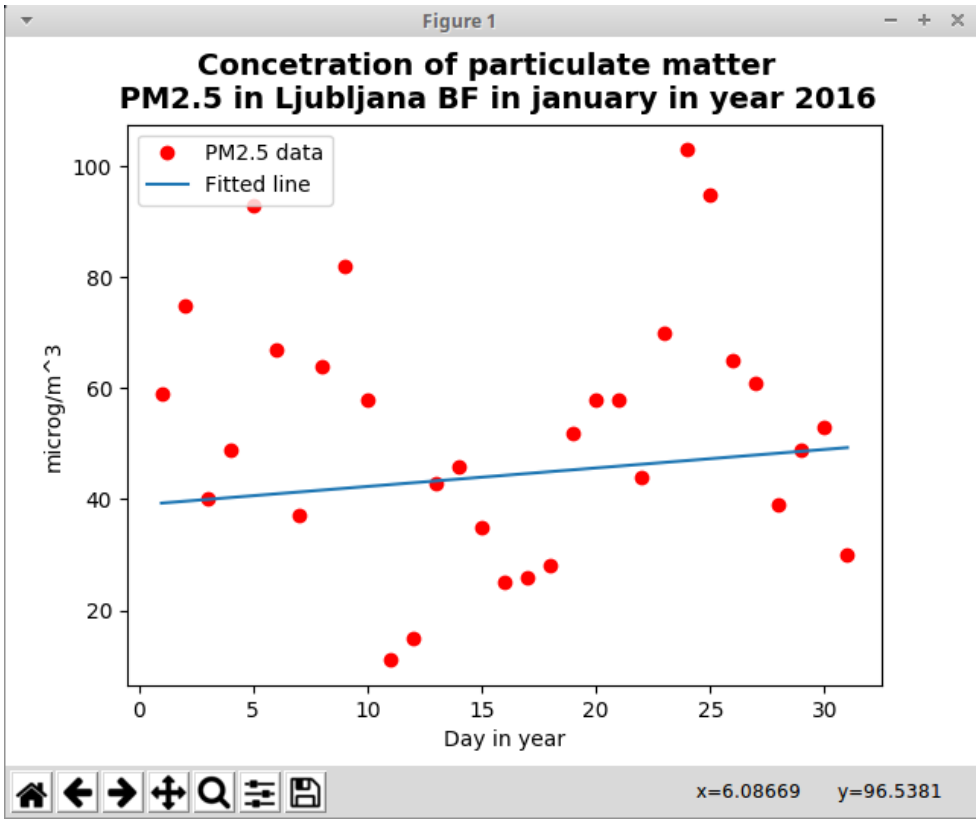
Grafični prikaz podatkov, pridobljenih z programom **slr.py**, se izvaja v programu imen. **display_results_slr.py**. Prikaz rezultatov se ne izvaja v programu **slr.py**, ker je ta program namenjen za uporabo na superračunalniku Rudolf, python paket **matplotlib**, potreben za prikaz rezultatov v obliki grafov, ni nameščen na superračunalniku. Zato se prikaz rezultatov izvaja lokalno na osebnem računalniku.

Kratek opis uporabe in poteka programa **display_results_slr.py**:

- Program poženemo v direktoriju, kjer se nahajajo z programom **slr.py** ustvarjene podatkovne datoteke **o_data_X.txt**, **o_data_Y.txt** in **o_fLine.txt**, v Linux terminalu z ukazom

```
$ python display_results_slr.py
```

- Program prebere omenjene podatkovne datoteke ter jih s python paketom **matplotlib** pretvori v graf, kot je prikazan v Slika 3.



Slika 3: Pimer izrisa grafa s programom display_results_slr.py

Program **display_results_slr.py**:

```
1   # Author: Dejan Penko
2   # This script is used to create graphs for data (o_data_*.txt output files)
3   # obtained by the simple linear regression "slr.py" code.
4   # Note that correct graph options (title, axis labels) have to be set
5   # by the user.
6
7   import numpy as np
8   import matplotlib.pyplot as plt
9
10  # Open .txt files and get the numpy arrays
11  data_X = np.loadtxt("o_data_X.txt", dtype=float)
12  data_Y = np.loadtxt("o_data_Y.txt", dtype=float)
13  fLine = np.loadtxt("o_fLine.txt", dtype=float)
14
15  # Graphic display
16  fig = plt.figure()
17  fig.suptitle( \
18      'Concetration of particulate matter \n PM2.5 in Ljubljana BF in january in \
19       year 2016', \
```

```
20      fontsize=14, fontweight='bold')
21  ax = fig.add_subplot(111)
22  ax.plot(data_X, data_Y, 'ro', label = "PM2.5 data")
23  ax.plot(data_X, fLine, label = "Fitted line")
24  ax.set_xlabel("Day in year")
25  ax.set_ylabel("microg/m^3")
26  ax.plot
27  ax.legend()
28  plt.show()
```

5 POSTOPEK IZVEDBE NALOGE NA SUPERRAČUNALNIKU RUDOLF

V tem poglavju je prikazan le princip izvedbe omenjene naloge na superračunalniku. Dostop do superračunalnika je obrazložen v poročilu **Superračunalnik_za_vse-Namestitev_in_uporaba**.

Za pošiljanje naloge na superračunalnik potrebujemo:

- program **slr.py**
- podatkovno .txt datoteko (v tem primeru PM2.5_D_dec16_ang.txt)
- datoteko **slr.sh**
- datoteko **slr.xrsl**

V datoteki **slr.sh** je zapisan ukaz, kateri se izvede na superračunalniku:

Vsebina datoteke **slr.sh**:

```
#!/bin/sh
python slr.py -i PM2.5_D_dec16_ang.txt -c 2 -m jan
```

V datoteki **slr.xrsl** pa so definirane:

- datoteka z ukaznimi vrsticami **slr.sh**,
- vhodne datoteke **PM2.5_D_dec16_ang.txt** in **slr.py**,
- izhodne datoteke **output.txt**, **o_data_X.txt**, **o_data_Y.txt** in **o_fLine.txt**,
- ime naloge **PKPsimLinReg** ter
- okolje v superračunalniku Rudolf **APPS/BASE/TENSORFLOW**

Vsebina datoteke **slr.xrsl**:

```
&
(executable="slr.sh")
(inputfiles=
("PM2.5_D_dec16_ang.txt" "PM2.5_D_dec16_ang.txt")
("slr.py" "slr.py")
)
(outputfiles=
("output.txt"  "output.txt")
("o_data_X.txt"  "o_data_X.txt")
("o_data_Y.txt"  "o_data_Y.txt")
("o_fLine.txt"  "o_fLine.txt")
)
(stdout="out.txt")
(stderr="err.txt")
(gmlog="gmlog.log")
(jobName="PKPsimLinReg")
(runtimeenvironment = "APPS/BASE/TENSORFLOW")
```

Nalogo pošljemo v sistem z ukazom

```
$ arcsub -c jost.arnes.si -o joblist.xml slr.xrsl
```

Ter preverjamo status naloge z ukazom

```
$ arcstat gsiftp://jost.arnes.si:2811/jobs/<vpišite-ID-tukaj>
```

Ko je naloga na superračunalniku zaključena prenesemo rezultate z ukazom

```
$ arcget gsiftp://jost.arnes.si:2811/jobs/<vpišite-ID-tukaj>
```

ter prikažemo rezultate z uporabo **display_results_slr.py** programa znotraj direktorija, kateri vsebuje prenešene datoteke z rezultati

```
$ python display_results_slr.py
```

6 ZAKLJUČEK

Sam postopek priprave podatkov, izračuna linearne regresije na superračunalniku ter prikaz rezultatov ni enostaven oz. zahteva več predhodnega znanja ter izkušenj iz programiranja ter dela s superračunalnikom. V kolikor pa imamo vse potrebno znanje, je superračunalnik uporabno orodje za obdelavo podatkov ter izvedbo raznovrsnih izračunov, kateri bi na osebnem računalniku potrebovali občutno več časa.

Tako smo s predhodno pripravo podatkov ter pripravo programa za izračun s pomočjo superračunalnika prišli do obdelanih podatkov, katere smo na koncu še grafično prikazali ter tako prišli do končnih rezultatov, ta postopek pa se lahko ponovi z uporabo drugih podatkov.

7 LITERATURA IN VIRI

http://e-lin.github.io/wiki/jekyll/update/2016/04/26/Forecasting-Fine-Grained-Air-Quality-Based-on-Big-Data.html

https://www.tensorflow.org

https://pypi.python.org/pypi/tabula-py

https://blog.chezo.uno/tabula-py-extract-table-from-pdf-into-python-dataframe-6c7acfa5f302

https://matplotlib.org/