Домашнее задание к 7 уроку.

Дедлайн: 23.12.2020

colab: https://colab.research.google.com/drive/1KLD5pDCa0ka_g1U4D2ExihaaEwXzfTPW?usp=sharing

ФОРМАТ ОТЧЕТНОСТИ: pdf-файл с решенными задачами (задачи 2-9), ноутбук с проверкой решения этих задач (при помощи numpy), и pdf файл с отчетом по 1-ой задаче (краткий пересказ статьи). Вы можете оформить решения с использованием Markdown (писать текст прямо в юпитер ноутбуке), Latex, Word или же решить в тетради и сформировать pdf из фото.

Итого: 2 pdf файла.

Все задания необходимо выполнять ВРУЧНУЮ. А также проверить корректность полученных результатов с использованием Numpy.

Туториал как писать "Latex-формулы" прямо в Юпитере: https://www.youtube.com/watch?v=vSc25kdgecg

Ноутбук с примером формул:
https://nbviewer.jupyter.org/github/postlogist/course_opt/blob/master/jupyter_tutorial/02_markdown.ipynb

Пример

Найдем ранг матрицы

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 8 & 11 & 14 \\ 3 & 9 & 14 & 20 & 26 \\ 5 & 14 & 22 & 31 & 40 \end{pmatrix}.$$

Четвертая строка является суммой второй и третьей строк, а значит, ее можно отбросить:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 8 & 11 & 14 \\ 3 & 9 & 14 & 20 & 26 \end{pmatrix}.$$

Из второй и третьей строк вычтем первую, умноженную на 2 и 3 соответственно:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 5 & 8 & 11 \end{pmatrix}.$$

И вычтем из третьей строки вторую, умноженную на 2:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 & 3 \end{pmatrix}.$$

```
# Проверка

import numpy as np
a = [1, 2, 3, 4, 5]
b = [2, 5, 8, 11, 14]
c = [3, 9, 14, 20, 26]
d = [5, 14, 22, 31, 40]

x = np.array([a, b, c, d])
r = np.linalg.matrix_rank(x)

print(f'Ранг матрицы: {r}')
```

```
Ранг матрицы: 3
```

Задачи

1. Прочитать статью http://www.scielo.org.mx/pdf/cys/v18n3/v18n3a7.pdf и сделать следующее:

- Написать кратко (не более 300 слов (минимум 100), отчет в формате pdf) о различиях между cosine similarity и soft similarity. Привести примеры использования и написать собственный пример вычисления cosine similarity и soft similarity для произвольных векторов (не брать вектора из статьи! надо самим придумать координаты/размерность векторов)

Cosine similarity является мерой вычисления сходства между двумя векторами. Фактически вычисляется косинус угла между векторами, отображая сходства направленности. Чем меньше угол тем больше косинус и при нулевом значении угла косинус равен 1, что соответствует полной схожести значений. Сходство косинуса рассматривает векторы как изначально независимые друг от друга и расчитывается как произведение веркторов разделенное на произведение норм. Для обработки текста такой подход не совсем подходит т.к. не учитывает контекст слов и их семантическое значение. А soft cosine similarity дополнительно учитывает сходство для пары слов с помощью создания матрицы, вычесленной по расстоянию Левенштейна. И она перемножается в тойже формуле, добавляя семантическую связь в измерение. Если между элементами вектора отсутствует сходство, то результат будет тот же, что и у обычного косинусного сходства.

```
from gensim.matutils import softcossim
from gensim import corpora
import gensim.downloader as api
```

```
#создание матрицы схожести по словарю
rus = api.load('word2vec-ruscorpora-300')
sent_1 = 'Я пытаюсь разобраться в создании мягкого сходства и векторах'.lower().split()
sent_2 = 'Я только и делаю вид и уже не пытаюсь разобраться в создании сходства'.lower().s
doc=[sent_1, sent_2]
dictionary = corpora.Dictionary(doc)
similarity_matrix = rus.similarity_matrix(dictionary)

#теперь необходимо выявить количество уникальных слов для определения пространства
sent_unique_1 = []
for word in sent_1:
    if word not in sent_unique_1:
        sent_unique_1.append(word)

sent_unique_2 = []
for word in sent_2:
    if word not in sent_unique_2:
        sent_unique_2.append(word)

documents = sent_unique_2 + sent_unique_1
documents = sorted(list(set(documents)))
for word in documents:
  if word == 'в' or 'и':
    documents.remove(word)
#7 уникальных слов
#теперь необходимо создать векторы в этом пространстве

vec=[]
for i in documents:
  if i in sent_1:
    count=sent_1.count(i)
    vec.append((documents.index(i),count))
  else:
    vec.append((documents.index(i), 0))
vec1=[]
for i in documents:
  if i in sent_2:
    count=sent_2.count(i)
    vec1.append((documents.index(i),count))
  else:
    vec1.append((documents.index(i), 0))

# векторы есть, теперь необходимо расчитать soft cosine similarity
# с использованием матрицы
print(softcossim(vec1, vec, similarity_matrix))
```

```
# я если честно не совсем понимаю логику наполнения векторов
# как вариант еще можно посчитать частоту встречаемости во всем тексте
all_text=sent_1+sent_2
vec2=[]
for i in sent_1:
  count=all_text.count(i)
  vec2.append((sent_1.index(i),count))
vec3=[]
for i in sent_2:
  count=all_text.count(i)
  vec3.append((sent_2.index(i),count))
print(softcossim(vec2, vec3, similarity_matrix))
```

2. Найти сумму и произведение матриц $A = \begin{pmatrix} 1 & -2 \\ 3 & 0 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & -1 \\ 0 & 5 \end{pmatrix}$.

3. Из закономерностей сложения и умножения матриц на число можно сделать вывод, что матрицы одного размера образуют линейное пространство. Вычислить линейную комбинацию $3A - 2B + 4C$ для матриц $A = \begin{pmatrix} 1 & 7 \\ 3 & -6 \end{pmatrix}$, $B = \begin{pmatrix} 0 & 5 \\ 2 & -1 \end{pmatrix}$, $C = \begin{pmatrix} 2 & -4 \\ 1 & 1 \end{pmatrix}$.

4. Дана матрица $A = \begin{pmatrix} 4 & 1 \\ 5 & -2 \\ 2 & 3 \end{pmatrix}$. Вычислить AA^T и A^TA.

5*. Написать на Python функцию для перемножения двух произвольных матриц, не используя NumPy.

6. а) $\begin{pmatrix} sinx & -cosx \\ cosx & sinx \end{pmatrix} = sinx^2 - (-cosx)2 = 1$

6. б)

$$\begin{pmatrix} 8 & 4 & 6 \\ 0 & 5 & 1 \\ 0 & 0 & 9 \end{pmatrix} = 8*5*9+0*4*1+0*0*6-0*5*6-8*0*1-0*5*9 =$$

6. б)

$$\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} = 2*6*10+8*3*7+5*9*4-8*6*4-2*9*7-5*3*$$
$$-150 = 468-468 = 0$$

6. Вычислить определитель (используйте любой удобный для вас способ вычисления определителя: через миноры, через перестановки или другой):

а)

$$\begin{vmatrix} sinx & -cosx \\ cosx & sinx \end{vmatrix};$$

б)

$$\begin{vmatrix} 8 & 4 & 6 \\ 0 & 5 & 1 \\ 0 & 0 & 9 \end{vmatrix};$$

в)

$$\begin{vmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{vmatrix}.$$

7. Определитель матрицы A равен 4. Найти:

а) $det(A^2)$;

б) $det(A^T)$;

в) $det(2A)$.

8. Доказать, что матрица

$$\begin{pmatrix} -2 & 7 & -3 \\ 4 & -14 & 6 \\ -3 & 7 & 13 \end{pmatrix}$$

вырожденная.

9. Найти ранг матрицы:

а) $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 3 & 4 \end{pmatrix}$;

б) $\begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 4 & 3 \\ 2 & 3 & 5 & 6 \end{pmatrix}$.

2.Сложение $\begin{pmatrix} 1 & -2 \\ 3 & 0 \end{pmatrix} + \begin{pmatrix} 4 & -1 \\ 0 & 5 \end{pmatrix} = \begin{pmatrix} 5 & -3 \\ 3 & 5 \end{pmatrix}$

2. Умножение

$$\begin{pmatrix} 1 & -2 \\ 3 & 0 \end{pmatrix} * \begin{pmatrix} 4 & -1 \\ 0 & 5 \end{pmatrix} = \begin{pmatrix} 1*4+(-2)*0 & 1*(-1)+(-2)*5 \\ 3*4+0*0 & 3*(-1)+0*5 \end{pmatrix} = \begin{pmatrix} 4 & - \\ 12 & \end{pmatrix}$$

```
a = [1, -2]
b = [3, 0]
x = np.array([a, b])

a = [4, -1]
b = [0, 5]


y = np.array([a, b])
print(x+y)

print(x.dot(y))
```

```
[[ 5 -3]
 [ 3  5]]
[[  4 -11]
 [ 12  -3]]
```

3. $$3\begin{pmatrix} 1 & 7 \\ 3 & 6 \end{pmatrix} - 2\begin{pmatrix} 0 & 5 \\ 2 & -1 \end{pmatrix} + 4\begin{pmatrix} 2 & -4 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 21 \\ 9 & 18 \end{pmatrix} - \begin{pmatrix} 0 & 10 \\ 4 & -2 \end{pmatrix} + \begin{pmatrix} 8 & -1 \\ 4 & 4 \end{pmatrix}$$

$$= \begin{pmatrix} 3-0+8 & 21-10-16 \\ 9-4+4 & 18+2+4 \end{pmatrix} = \begin{pmatrix} 11 & -5 \\ 9 & 24 \end{pmatrix}$$

```
a = [1, 7]
b = [3, 6]
x = np.array([a, b])

a = [0, 5]
b = [2, -1]

y = np.array([a, b])

a = [2, -4]
b = [1, 1]

z = np.array([a, b])
lin_comb=3*x-2*y+4*z
lin_comb
```

```
array([[11, -5],
       [ 9, 24]])
```

4.

$$AA^T = \begin{pmatrix} 4 & 1 \\ 5 & -2 \\ 2 & 3 \end{pmatrix} * \begin{pmatrix} 4 & 5 & 2 \\ 1 & -2 & 3 \end{pmatrix} = \left(\begin{matrix} 4^2 + 1^2 & 4 * 5 + 1 * (-2) & \\ 5 * 4 + (-2) * 1 & 5^2 + (-2)^2 & 5 \\ 2 * 4 + 3 * 1 & 2 * 5 + 3 * (-2) & \end{matrix} \right.$$

$$= \begin{pmatrix} 17 & 18 & 11 \\ 18 & 29 & 4 \\ 11 & 4 & 31 \end{pmatrix}$$

```
a = [4, 5, 2]
b = [1, -2, 3]
x = np.array([a, b])

a = [4, 1]
b = [5, -2]
c = [2, 3]

y = np.array([a, b, c])
y.dot(x)
```

```
array([[17, 18, 11],
       [18, 29,  4],
       [11,  4, 13]])
```

4.

$$A^T A = \begin{pmatrix} 4 & 5 & 2 \\ 1 & -2 & 3 \end{pmatrix} * \begin{pmatrix} 4 & 1 \\ 5 & -2 \\ 2 & 3 \end{pmatrix} = \left(\begin{matrix} 4^2 + 5^5 + 2^2 & 4 * 1 + 5 * (- \\ 1 * 4 + (-2) * 5 + 3 * 2 & 1^1 + (-2 \end{matrix} \right.$$

```
a = [4, 5, 2]
b = [1, -2, 3]
x = np.array([a, b])

a = [4, 1]
b = [5, -2]
c = [2, 3]

y = np.array([a, b, c])
x.dot(y)
```

```
array([[45,  0],
       [ 0, 14]])
```

```
# Задание №5
A= [[1, 3, 5, 5],
    [2, 6, 6, 6],
    [4, 8, 8, 2],
    [3, 1, 4, 5],
```

```
     [5, 1, 6, 7]]
B= [[2, 4, 5, 9, 4],
     [2, 7, 9, 3, 8],
     [3, 6, 2, 7, 3],
     [2, 8, 9, 9, 4]]



def martix(A,B):
  if len(A[0]) != len(B):
    print('''Умножать матрицы можно тогда и только тогда,
когда количество столбцов первой матрицы равно количеству строк второй матрицы''')
  else:
#сначала создаем пустую матрицу, по которой можно итерировать
    row=0
    column=0
    C=[]
    for i in A:
      C.append([])
    for i in C:
      while len(i)<len(A):
        j = 0
        i.append(j)
    row_a=0
    column_a=0
    row_b=0
    column_b=0
    elem=0
#теперь необходимо по каждому элементу провести операции сложения произведений
    for i in C:
      while elem<len(C):
          while row_b<len(B):
            i[elem]+=(A[row_a])[column_a]*(B[row_b])[column_b]
            column_a+=1
            row_b+=1
          elem+=1
          row_b=0
          column_a=0
          column_b+=1
      row_a+=1
      column_a=0
      row_b=0
      column_b=0
      elem=0

      row_b=0
    return C


  #return C
martix(A,B)
```

```
 [[33, 95, 87, 98, 63],
  [46, 134, 130, 132, 98],
```

```
       [52, 136, 126, 134, 112],
       [30, 83, 77, 103, 52],
       [44, 119, 109, 153, 74]]
```

6. а) $\begin{pmatrix} sinx & -cosx \\ cosx & sinx \end{pmatrix} = sinx^2 - (-cosx)2 = 1$

```
import numpy as np
x=np.random.sample()
a = [np.sin(x), -(np.cos(x))]
b = [np.cos(x), np.sin(x)]


x = np.array([a, b])
r=np.linalg.det(x)

r=float('{:.6f}'.format(r))
print(r)
```

```
1.0
```

6. б)

$$\begin{pmatrix} 8 & 4 & 6 \\ 0 & 5 & 1 \\ 0 & 0 & 9 \end{pmatrix} = 8*5*9+0*4*1+0*0*6-0*5*6-8*0*1-0*5*9=$$

```
a = [8, 4, 6]
b = [0, 5, 1]
c = [0, 0, 9]

x = np.array([a, b, c])
r=np.linalg.det(x)
r=float('{:.6f}'.format(r))
print(r)
```

```
360.0
```

6. б)

$$\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix} = 2*6*10+8*3*7+5*9*4-8*6*4-2*9*7-5*3*$$

$$-150 = 468 - 468 = 0$$

```
a = [2, 3, 4]
b = [5, 6, 7]
c = [8, 9, 10]
```

```
x = np.array([a, b, c])
r=np.linalg.det(x)
r
```

```
0.0
```

7. а) $det(A^2) = |A| * |A| = 4 * 4 = 16$;

б) $det(A^T) = 4$;

в) $det(2A) = 8$.

8. $$\begin{pmatrix} -2 & 7 & -3 \\ 4 & -14 & 6 \\ -3 & 7 & 13 \end{pmatrix} = (-2) * (-14) * 13 + (-3) * 7 * 6 + 4 * 7 * (-3) - (-3) *$$
$$* 7 * 13 = 364 + (-126) + (-84) - (-126) - (-84) - 364 = 0$$
Определитель матрицы равен 0, а значит матрица вырожденная

```
a = [-2, 7, -3]
b = [4, -14, 6]
c = [-3, 7, 13]

x = np.array([a, b, c])
r=np.linalg.det(x)
if r == 0:
  print('матрица вырожденная')
else:
  print('матрица не вырожденная')
```

```
матрица вырожденная
```

9. из первой строки вычтем вторую, затем из третьей строки вычтем две вторых строки. Т.к. строки 1 и 3 равны убираем последнюю и матрица становится 2 на 3. Следовательно ранг равен 2.
$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 3 & 4 \end{pmatrix} -- \begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} -- \begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

```
import numpy as np
a = [1, 2, 3]
b = [1, 1, 1]
c = [2, 3, 4]


x = np.array([a, b, c])
r = np.linalg.matrix_rank(x)
```

```
print(f'Ранг матрицы: {r}')
```

```
Ранг матрицы: 2
```

9. К первой строке прибавляем третью и вычетаем из не три вторых строки. Потом третью строку умножаем на два и вычитваем из нее четыре вторых строки. Т.к. первая и третья строки эквивалентны удаляем первую и получаем матрицу из трех независимых строк. Значит ранг равен 3
$$\begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 4 & 3 \\ 2 & 3 & 5 & 6 \end{pmatrix} -- \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 4 & 3 \\ 2 & 3 & 5 & 6 \end{pmatrix} -- \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 6 \end{pmatrix} -- \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 2 & 3 & 5 & 6 \end{pmatrix}$$

```
import numpy as np
a = [0, 0, 2, 1]
b = [0, 0, 2, 2]
c = [0, 0, 3, 4]
d = [2, 3, 5, 6]

x = np.array([a, b, c, d])
r = np.linalg.matrix_rank(x)

print(f'Ранг матрицы: {r}')
```

```
Ранг матрицы: 3
```

Доп материалы

1. [Способы задать матрицу в NumPy](#).
2. [numpy.transpose](#).
3. [array.T](#).
4. [Перемножение матриц в NumPy](#).
5. [Определитель матрицы в NumPy](#)
6. [Ранг матрицы в NumPy](#)
7. [Обращение матриц в NumPy](#)