ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Линейная алгебра с библиотекой Numpy.

Аннотация к занятию: В первой части урока обучающиеся изучают новые понятия, изучают основные операции с векторами в Numpy. Во второй части обучающиеся решают задачи.

Цель занятия: формирование у учеников представления о линейной алгебра с библиотекой Numpy.

Задачи занятия:

1. познакомить с линейной алгеброй с библиотекой numpy;
2. рассмотреть основные математические операции по работе с векторами numpy.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	5 мин.	Добрый день! Мы продолжаем изучать библиотеку Numpy	Приветствие. Создание в классе атмосферы психологическог о комфорта.
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	7 мин.	**Вопрос для обсуждения:** Чем вектор отличается от матрицы? **Ответы школьников:** Вектор – одномерный, матрица двумерная. **Знакомство с линейной алгеброй** Вы уже знаете, что данные записываются и хранятся в разных форматах. Для машинного обучения обычное дело — проводить математические операции с наборами данных. Поэтому все исходные значения превращаются в математические объекты. Можно ли заниматься машинным обучением без математики? Если искать в интернете уже готовые алгоритмы и всегда использовать библиотеки, то математика не нужна. Так можно решить поставленную задачу, но невозможно придумать что-то уникальное, решение, которое будет лучше других. Мы покажем, как работают настоящие профессионалы. Чтобы понять, как	Способствовать обсуждению мотивационных вопросов.

		устроена работа с массивами NumPy в машинном обучении, тебе нужны два понятия — векторы и матрицы. О них сейчас и поговорим.	
Изучение нового материала	50 мин.	**На основе ноутбука:** https://drive.google.com/file/d/1qstpsZSDR12ZmUZnPSOTJIdU-miz-Frg/view?usp=sharing Скорость операций Ранее мы уже обсуждали преимущество в скорости, которое дает библиотека numpy. Давайте убедимся в этом на практике. В примере создадим 100000000 элементов и добавим к ним десятку с помощью чистого питона и генератора спсиков и с помощью numpy. Если к numpy массиву прибавляют (или производят любую другую мат операцию) элемент, то он прибавляется ко всем элементам. Такое свойство назвают по русски вещанием (перевод от broadcasting). Разница в скорости ощутима - numpy потребовалось миллисекунды, тогда как у питона на это ушло аж 15 секунд.	

```python
import datetime
start = datetime.datetime.now()

a = [1 for i in range(100000000)]
a = [elem + 10 for elem in a]

end = datetime.datetime.now()
(end-start).seconds
```

```
15
```

```python
import datetime
import numpy as np
start = datetime.datetime.now()

a = np.arange(100000000)
a = a+10

end = datetime.datetime.now()
(end-start).seconds
```

```
0
```

Векторные операции

Подобная процедура вещания, т.е. применение простейших математических преобразований поэлементно может работать не только между нампай массивом и скаляром. В примере ниже значения с одинаковыми индексами умножаются друг на друга, порождая новый массив:

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

```
[ ]  1  a * b
```

```
array([ 3,  8, 15])
```

Для того, чтобы задействовать функционал из линейной алгебры необходимо использовать один из двух подходов - или с помощью символа собаки @ или с помощью метода dot (от англ. dot product, скалярное произведение). В примере мы находим скалярное произведение двух массивов:

```
[ ]  1  a @ b
```

```
26
```

```
1  np.dot(a, b)
```

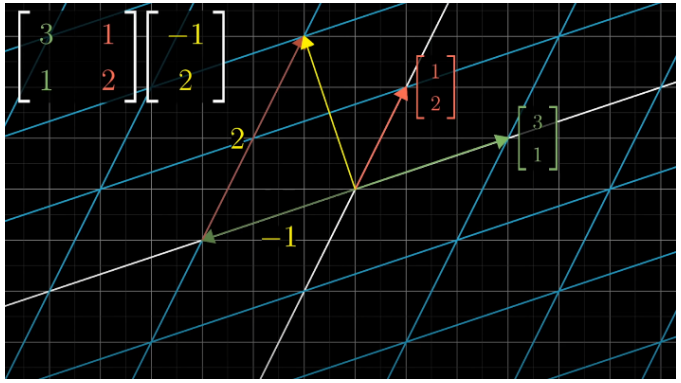
```
26
```

По аналогии работают и операции с двумерными массивами (матрицами). Использование математических символов приводит к поэлементным операциям (т.е. операциям со значениями на одинаковых индексах):

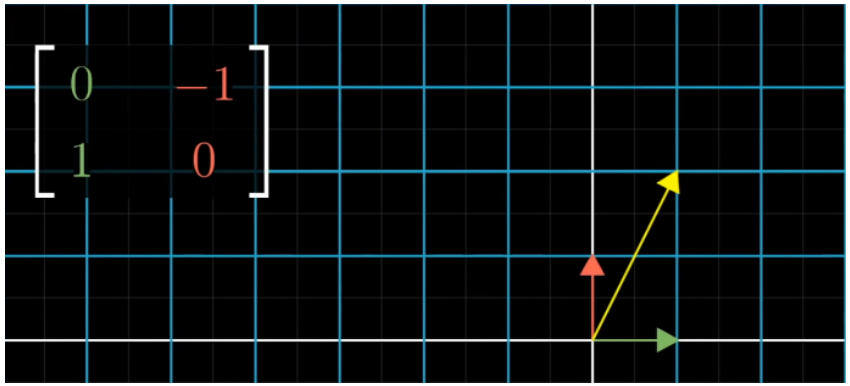
```
1  a = np.array([[3, 1], [1, 2]])
2  b = np.array([[1, 3], [2, 4]])
3
4  a * b
```

```
array([[3, 3],
       [2, 8]])
```

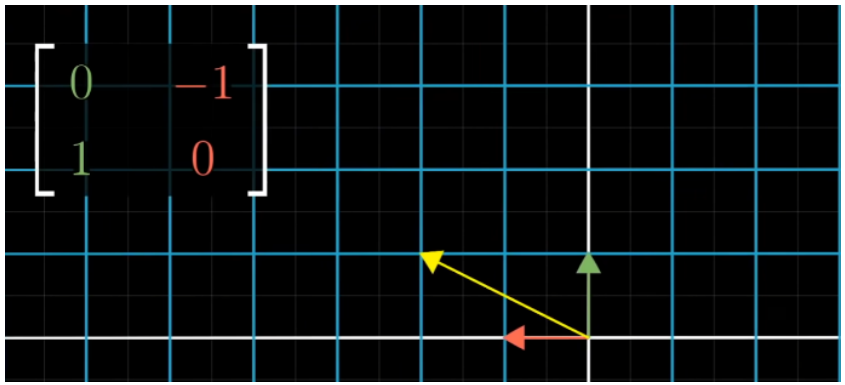
Использование же @ или dot вызовет векторно-матричную, матрично-векторную или метрично-матричную операцию.
В примере ниже мы умножаем матрицу b (3, 1; 1, 2) на вектор a (-1, 2). Сама эта операция может быть легко визуализирована. Зеленым и красным соответственно показано куда будут отображены (базисные) вектора i (1, 0) и j (0, 1). Зная, куда отобразятся единичные (базисные) вектора мы можем понять как отразится вектор a (как и любая другая точка пространства в этом линейном преобразовании, на схеме это показано синими и белыми линиями):



Аналогичный пример с матрицей поворота:



Вектор повернулся влево на 90 градусов:



Эта же операция в numpy:

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

```
      1   a @ b
```

```
array([-2,  1])
```

Инициализация и свойства матриц

На прошлом занятии мы уже рассматривали инициализацию пустой матрицы, из единиц и нулей. Так можно задать матрицу из 10 элементов с заданным нулевым средним и 0.1 стандартным отклонением:

```
[ ]   1   mu, sigma = 0, 0.1 # среднее и стандартное отклонение
      2   np.random.normal(mu, sigma, 10)
```

```
array([ 0.05182366,  0.01484232,  0.13333666, -0.03161026, -0.03531011,
       -0.04523209,  0.11514029, -0.11188967,  0.01920465,  0.13850302])
```

Задавать значения из равномерного распределения можно в диапазоне 0 на 1 с фиксированной формой из 3 строк и 2 столбцов:

```
      1   np.random.rand(3,2)
```

```
array([[0.4621139 , 0.42726163],
       [0.45515588, 0.24764275],
       [0.55477837, 0.88097922]])
```

Или задать случайные значения от 0 до 3 из десяти строк и 2 столбцов:

```
1  np.random.randint(3, size=(10, 2))
```

```
array([[2, 0],
       [1, 0],
       [0, 0],
       [0, 1],
       [0, 1],
       [1, 0],
       [0, 0],
       [2, 2],
       [0, 2],
       [2, 2]])
```

+ К

Операции линейной алгебры

Создадим матрицу формы 3 на 3 из значений от 0 до 9:

```
[ ]  1  a = np.arange(9).reshape(3, 3)
     2  a
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Встроенный метод .T позволяет транспонировать матрицу:

```
[ ]   1   a.T
```

```
array([[0, 3, 6],
       [1, 4, 7],
       [2, 5, 8]])
```

Ранее мы импортировали функционал подмодуля для работы с линейной алгеброй в переменную LA:

```
[ ]   1   from numpy import linalg as LA
```

С ее помощью можно находить норму матрицы (по умолчанию L2 норма, но можно использовать и другие, напр. Фробениуса):

```
[ ]   1   LA.norm(a)
```

```
14.2828568570857
```

В модуле линалг (LA) можно найти множество полезных для области машинного обучения (и не только) функций. Например он позволяет решать систему линейных уравнений:

Рассмотрим систему уравнений:

$$\begin{cases} x_0 + 2x_1 - 3x_2 = 4 \\ 2x_0 + x_1 + 2x_2 = 3 \\ 3x_0 - 2x_1 - x_2 = 9 \end{cases}$$

Мы можем переписать их как матрицу коэффициентов возле x_1, x_2, x_3 и вектор-столбец ответов.

```
[ ]  1  a = np.array([[1, 2, -3],
     2                [2, 1, 2],
     3                [3, -2, -1]])
     4
     5  b = np.array([4, 3, 9])
```

Numpy позволяет решать системы линейных уравнений:

```
[ ]  1  x = LA.solve(a, b)
     2  x
```

```
array([ 2.475, -0.4  , -0.775])
```

Или, например, находить обратную матрицу:

```
[ ]  1  a = np.random.randint(10, size = (6, 6))
     2  a
```

```
array([[0, 1, 2, 9, 8, 7],
       [1, 1, 0, 1, 5, 6],
       [7, 8, 6, 9, 3, 3],
       [7, 1, 4, 2, 0, 4],
       [1, 6, 5, 1, 4, 8],
       [9, 4, 6, 6, 0, 7]])
```

```
[ ]  1  inv = np.linalg.inv(a)
     2  inv
```

```
array([[-0.07451836,  0.15519032,  0.07189593,  0.12237109, -0.09887314,
        -0.04624292],
       [-0.12468193,  0.13213377,  0.09869138, -0.24263904,  0.01017812,
         0.09614686],
       [ 0.13522356, -0.34467986, -0.03185854,  0.38456146,  0.18429662,
        -0.25650413],
       [ 0.0694293 , -0.05504492, -0.00810095, -0.16871787, -0.06397674,
         0.15075038],
       [ 0.03598691,  0.07244119,  0.10500078,  0.30222776,  0.0087241 ,
        -0.32575167],
       [-0.0083606 ,  0.06758581, -0.11458171, -0.20369216,  0.01817521,
         0.23801734]])
```

Этот функционал, реализующий быстрые алгоритмы вычисления обратной матрицы очень пригодится нам в ходе освоения материала линейной регрессии.
Всегда можно проверить, что результат умножения исходной матрицы на обратную (с точностью до округления в регистрах) дает нам единичную матрицу.

Закрепление изученного материала	15 мин.	Повторим функции Numpy, установите соответствие между функцией и ее описанием https://learningapps.org/view6919395 Поделимся на 2 группы: 1 вариант решает задачу 1, 2 вариант – задачу 2. **Задача 1** Точка где находится наш квадрокоптера A=(35,17) Робот двигается так: 1. Летит 14 шагов по вектору (-1,1). 2. Летит 3 шага по вектору (-1,0) 3. Повторяет еще раз первые два шага. 4. Садится. Выведите координаты посадки нашего квадрокоптера. position=np.array([35,17]) **Задача 2** У нас есть некая матрица с числами от 0 до 255, n строк k ячеек в строке. 1. Посчитайте сумму значений главной диагонали. 2. Объясните, в каком контексте это происходит и какой смысл этой операции? A = np.array([[1,-5,3],[2,2,1],[0,3,1],[2,4,12]])	Переходим по ссылке и выполняем все вместе задание. https://learningapps.org/view6919395 Деление на группы, решение и проверка задач
Этап подведения итогов занятия (рефлексия)	8 мин.	- Что вам на уроке больше всего понравилось? - С какими трудностями вы столкнулись?	Педагог способствует размышлению обучающихся над вопросами.

Информация о домашнем задании, инструктаж по его применению	5 мин.	В домашнем задании вам предстоит самостоятельно реализовать линейно-алгебраические операции с помощью чистого Python, а затем попрактиковаться в использовании уже готовых реализаций из numpy. Кроме того, вы сравните время работы вашей реализации с реализацией из библиотеки NumPy и убедитесь, что скорость работы "коробочных" реализаций выше.	Педагог инструктирует обучающихся о домашнем задании.

Рекомендуемые ресурсы для дополнительного изучения:

1. Знакомство с Numpy. [Электронный ресурс] – Режим доступа: https://proproprogs.ru/modules/numpy-ustanovka-i-pervoe-znakomstvo
2. Numpy: начало работы. [Электронный ресурс] – Режим доступа: https://pythonworld.ru/numpy/1.html
3. Numpy в Python. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/352678/
4. Учебник по Python Numpy. [Электронный ресурс] – Режим доступа: https://russianblogs.com/article/4050534552/