**Задания по курсу Deep Learning Gomel**

-----------------------------------------------------------------------------------------------------------------

Результаты заданий можно высылать на email’ы:

* nesterione@gmail.com,
* kaliostrogoblin3@gmail.com

или pull-request в репозиторий <https://github.com/nesterione/dl-exercises>

-----------------------------------------------------------------------------------------------------------------

**Содержание**

[Неделя 1, задание 1](#_9ls5aue4h3ip)……………………………………………………………………………….... 2

[Неделя 2, задание 2](#_o93mlqkm2vf4) ………………………………………………………………………………....2

[Неделя 3, задание 3](#_7sbwgmv1aetv)……………………………………………………………………………….... 4

[Неделя 4, задание 4](#_pivy0hfduxtu)……………………………………………………………………………….... 4

[Неделя 5, задание 5](#_1y482fp5fg4p)……………………………………………………………………………….... 4

[Неделя 6, задание 6](#_q0fbde9fqk7c)……………………………………………………………………………….... 5

[Неделя 7, задание 7](#_sl29wofakhvf)……………………………………………………………………………….... 5

[Неделя 8, задание 8](#_uziphkxmu6mt)……………………………………………………………………………….... 6

[Неделя 10, задание 9](#_9rlpp5w2gjqr)……………………………………………………………………………….. 6

[Неделя 11, задание 10](#_ncr6d6smpb4o)……………………………………………………………………………… 6

# Неделя 1, задание 1

- Настроить окружение (установить python 3.5 и jupyter)

- Загрузить notebook с лекции (команда ‘***jupyter notebook***’) и проверить примеры (‘***Shift + Enter’*** в блоке с кодом)

- Создайте свой ноутбук и поэкспериментируйте с функциями python, попробуйте реализовать известные вам алгоритмы, а также обычные циклы и ветвления.

**Ссылка на статью**, о том, как использовать jupyter notebook: <https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook>

# Неделя 2, задание 2

В основном репозитории курса, в папке lesson2 (<https://github.com/nesterione/dl/tree/master/lesson2>) расположены два файла: ***matrix.py*** и ***testMatrix.py***

Файл ***matrix.py*** содержит определение класса Matrix - класс матрицы - для которого переопределены методы:

* \_\_getitem\_\_ - индексация
* \_\_setitem\_\_ - назначение элемента по индексу.
* \_\_str\_\_ - вызывается функциями str, print и format. Возвращает строковое представление объекта.
* \_\_eq\_\_ - x == y вызывает x.\_\_eq\_\_(y)
* \_\_add\_\_ - сложение. x + y вызывает x.\_\_add\_\_(y)
* \_\_sub\_\_ - вычитание
* **\_\_**mul\_\_ - умножение (x \* y), произведение Адамара (см. презентацию)

Также класс содержит функции:

* getRank - получить размерность матрицы
* transpose - вернуть транспонированное представление матрицы
* dot - матричное перемножение
* fromList - создание матрицы из списка
* makeId - создание единичной матрицы

**Задание заключается в том, чтобы дописать функции:**

1. **transpose**
2. **dot**
3. **\_\_add\_\_**
4. **\_\_sub\_\_**

Для того, чтобы проверить правильность написанных функций, нужно скачать файл ***testMatrix.py***, который содержит тесты для класса Matrix, в ту же папку, что и файл с классом матрицы и запустить его. Если все тесты пройдены, то программа завершится без ошибок.

Вывод программы, в случае не прохождения тестов:

*FF..FF*

*======================================================================*

*FAIL: testAdd (\_\_main\_\_.MatrixTests)*

*----------------------------------------------------------------------*

*Traceback (most recent call last):*

*File "C:\Users\Pc\_Home\Google Диск\Deep Learning courses\dl\lesson2\testMatrix.py", line 10, in testAdd*

*self.assertTrue(m3 == Matrix.fromList([[8, 10, 12], [14,16,18]]))*

*AssertionError: False is not true*

*======================================================================*

*FAIL: testDot (\_\_main\_\_.MatrixTests)*

*----------------------------------------------------------------------*

*Traceback (most recent call last):*

*File "C:\Users\Pc\_Home\Google Диск\Deep Learning courses\dl\lesson2\testMatrix.py", line 30, in testDot*

*self.assertTrue(m1.dot(m2) == Matrix.fromList([[63, 69], [150, 165]]))*

*AssertionError: False is not true*

*======================================================================*

*FAIL: testSub (\_\_main\_\_.MatrixTests)*

*----------------------------------------------------------------------*

*Traceback (most recent call last):*

*File "C:\Users\Pc\_Home\Google Диск\Deep Learning courses\dl\lesson2\testMatrix.py", line 16, in testSub*

*self.assertTrue(m3 == Matrix.fromList([[6, 6, 6], [6, 6, 6]]))*

*AssertionError: False is not true*

*======================================================================*

*FAIL: testTranspose (\_\_main\_\_.MatrixTests)*

*----------------------------------------------------------------------*

*Traceback (most recent call last):*

*File "C:\Users\Pc\_Home\Google Диск\Deep Learning courses\dl\lesson2\testMatrix.py", line 38, in testTranspose*

*self.assertTrue(m1.transpose()==m2)*

*AssertionError: False is not true*

*----------------------------------------------------------------------*

*Ran 6 tests in 0.007s*

*FAILED (failures=4)*

Вывод программы верно прошедшей все тесты:

......

----------------------------------------------------------------------

Ran 6 tests in 0.000s

OK

[Finished in 0.5s]

Про перегрузку операторов можно почитать тут - <https://pythonworld.ru/osnovy/peregruzka-operatorov.html>. Про линейную алгебру в машинном обучении есть глава в книге Deep Learning <http://www.deeplearningbook.org/contents/linear_algebra.html>.

# Неделя 3, задание 3

Для выполнения домашнего задания необходимо следовать инструкциям в ноутбуке - <https://github.com/nesterione/dl/blob/master/lesson3/lesson3_ex.ipynb>

В ноутбуке каркас, где нужно дописать свой код в секции

*# PUT YOUR CODE HERE .*

Вы можете видеть промежуточные результаты, и по ним сравнивать. Если у вас возникают проблемы с перемножением матриц, смотрите их размерности (‘***array\_name.shape’***)

# Неделя 4, задание 4

Файлы, необходимые для выполнения четвертого задания, находятся в папке, в репозитории: <https://github.com/nesterione/dl/tree/master/lesson4/exercise>

1) требуется дополнить своим кодом методы класса ***LinearRegression***в файле "linear\_model.py"

в комментариях к коду можно найти подсказки, по каким формулам нужно составить алгоритм

2) требуется дописать методы: ***normalization, \_\_weight\_init, predict, cost*** и ***\_\_gradient***

3) в презентации к лекции объясняется алгоритм градиентного спуска для линейной регрессии

4) для проверки себя, в функции ***main*** вставлен код генерации данных для обучения. Если вы выполнили все правильно, то ошибка после выполнения алгоритма будет около 22

5) В ноутбуке "Linear Regression\_ex.ipynb" каркас, где нужно дописать свой код в секции:

### ТУТ ДОЛЖЕН БЫТЬ ВАШ КОД ###

Вы можете видеть промежуточные результаты, и по ним сравнивать.

**UPD:** В файле ошибка - нужно **сначала нормализовать входные параметры X, а потом добавлять единичный столбец**, потому что, если сделать наоборот, то при нормализации будет **деление на ноль**.

# Неделя 5, задание 5

Файлы задания найдёте здесь - <https://github.com/nesterione/dl/tree/master/lesson5/exercise>. Обратите внимание, что **это задание нужно выполнять после предыдущего**, тогда вам будет проще реализовать логистическую регрессию, т.к. большая часть уже будет готова с прошлого занятия.

**Порядок выполнения:**

1. Допишите код в файле linear\_model.py . Там есть main и вы можете запускать этот файл для проверки, обратите внимание, что скрипт почти такой же как и в предыдущем задании. Все новые задания помечены (**new**) в комментарии (3 шт).
2. После написания класса linear\_model приступайте к заданию в ноутбуке Logistic\_Regression\_ex - просто следуйте указаниям в файле.

Вам придётся решить классическую для машинного обучения задачу: **предсказание выживших после крушения Титаника** (<https://www.kaggle.com/c/titanic>) . Также, вы познакомитесь с платформой kaggle, где найдёте огромное количество задачек и конкурсов по машинному обучению.

**UPD:** В файле ошибка - нужно **сначала нормализовать входные параметры X, а потом добавлять единичный столбец**, потому что, если сделать наоборот, то при нормализации будет **деление на ноль**.

# Неделя 6, задание 6

1. Решить задачу о титанике (см. задание 5) с использованием SVM из scikit-learn, сравнить с логистической регрессией. (см. ссылки).

<http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

<https://www.kaggle.com/c/titanic>

**Провести подбор оптимальных параметров для классификатора (разные параметры регуляризации, и ядра для SVM)**

2) С использованием SVM реализовать классификатор чисел. Датасет использовать c kaggle <https://www.kaggle.com/c/digit-recognizer>

# Неделя 7, задание 7

1. На этот раз задача очень простая, возьмите соревнование на Kaggle с **распознаванием рукописных чисел** (<https://www.kaggle.com/c/digit-recognizer>) и попробуйте добиться максимально хорошего результата. Вы знаете несколько алгоритмов, и у вас в полном распоряжении библиотека scikit-learn.
2. 2) Отправьте свой ответ на платформу Kaggle и посмотрите, какой у вас результат в общем рейтинге. Для этого нужно зарегистрироваться и загрузить файл с ответами.
3. 3) Также сделайте ещё одно задание, в ноутбуке с лекции: <https://github.com/nesterione/dl/blob/master/lesson7/lesson7.ipynb>.

В разделе **Ансамбли алгоритмов** есть график - он показывает уровень ошибки, в зависимости от количества алгоритмов в ансамбле. Очевидно, что чем больше алгоритмов вы укажете, тем ниже ошибка будет, **НО** со временем это приведёт к переобучению. **Ваша задача**: на том же графике добавить ещё один, с проверкой ошибки на тестовой выборке. **Вам необходимо найти оптимальное количество алгоритмов, не приводящее к переобучению.**

# Неделя 8, задание 8

1. Запустить файл ноутбука из лекции: <https://github.com/nesterione/dl/blob/master/lesson8/Multilayer%20perceptron.ipynb>
2. поэкспериментировать с числом слоев, числом эпох и скоростью обучения (см. “***Construct neural network (init weights)”*** и “***Do 50 epochs of gradient descent”***). Важно понять, что происходит под капотом этой системы.
3. Отправить решение задачи MNIST на Kaggle (<https://www.kaggle.com/c/digit-recognizer>) и прислать мне kaliostrogoblin3@gmail.com скрин с результатом в таблице на Kaggle.
4. Вместе со скрином, напишите, какие параметры оказались более оптимальными:

* скорость обучения
* число слоев
* число нейронов на каждом слое
* число эпох
* стохастический градиентный спуск???

# Неделя 10, задание 9

Следуйте указаниям в ноутбуке: <https://github.com/nesterione/dl/tree/master/lesson10/exercise>

В ходе задания вы построите свою первую (или не первую) глубокую нейросеть с помощью Keras и обучите ее для выполнения задачи классификации для базы данных Cifar10 (<https://www.cs.toronto.edu/~kriz/cifar.html>).

# Неделя 11, задание 10

Обучить **сверточную нейронную сеть** для набора данных CIFAR10 <https://www.cs.toronto.edu/~kriz/cifar.html> . **Применить transfer learning**, за основу использовать нейронную сеть обученную на Imagenet.

Ход выполнения:

1. выбрать архитектуру сети, подходящую для применения transfer learning
2. найти веса сети обученной на наборе данных imagenet
3. применить transfer learning для выбранной архитектуры для задачи классификации на наборе данных CIFAR10
4. сравнить скорость обучения и accuracy (после 10-ти эпох обучения) с применением transfer learning и без.

**Ожидаемые результаты**:

С применением transfer learning ожидается выше точность за меньшее время обучения.