НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Системи штучного інтелекту

Методичні вказівки для виконання практичних робіт | Осінній семестр

Штучний інтелект

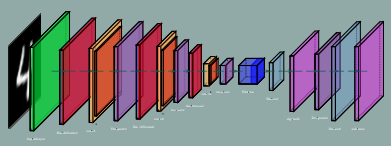
Будь-яка техніка, яка дозволяє комп’ютерам імітувати поведінку людини

Машинне навчання

Можливість комп’ютера учитися не будучи явно запрограмованим

Глибинне навчання

Пошук шаблону в даних за допомогою нейронних мереж



Лектор

СТІРЕНКО Сергій Григорович

Доктор технічних наук, професор

Кафедра ОТ, ФІОТ

sergii.stirenko@gmail.com

Асистент

КОЧУРА Юрій Петрович

Кафедра ОТ, ФІОТ

@y_kochura

iuriy.kochura@gmail.com

Особливості

- Для магістрів 1-го курсу, осінь
- 123 – “Комп’ютерна інженерія”
- Нормативна
- Очна форма навчання
- Українська, англійська
- 4 кредити ЄКТС
- 18 лекцій
- 4 практичні роботи + проєкт
- Екзамен

Опис

Цей курс познайомить Вас з підходами, які лежать в основі машинного та глибинного навчання та дозволить отримати практичний досвід:

- Використання нейронних мереж (повноз’єднані та згорткові шари, пряме та зворотне поширення, активаційні функції, ...)
- Тренування нейронних мереж (ініціалізація, оптимізація, регуляризація, вибір моделей, ...)

Потрібні навички

Для проходження цього курсу потрібно володіти наступними навичками:

- Рівень володіння англійською мовою не нижче A2.
- Знання Python на рівні, достатньому для написання нетривіального коду.
- Базові знання з математичної статистики, лінійної алгебри та теорії ймовірностей.

Система оцінювання

40%	Практичні завдання (10% кожне)
30%	Проєкт
30%	Екзамен

Важливо! Умова допуску до семестрового контролю (екзамену):

$$\text{Практичні завдання} + \text{Проєкт} \geq 42\%$$

Шкала оцінок КПІ ім. Ігоря Сікорського:

A = 95–100	Відмінно
B = 85–94	Дуже добре
C = 75–84	Добре
D = 65–74	Задовільно
E = 60–64	Достатньо
F < 60	Незадовільно
Fx < 42	Недопущений
Порушення кодексу честі	Усунений

Кодекс честі

Ви можете обговорювати завдання практичних робіт у групах. Однак, кожен студент/студентка повинен/повинна підготувати розв’язки завдань самостійно.

Під час проходження цього курсу Ви зобов’язані дотримуватись Кодекс честі КПІ ім. Ігоря Сікорського та усі наступні правила:

1. Кожен з Вас повинен відправляти на перевірку власно виконану роботу. Використання чужих розв’язків або програмного коду і представлення їх за свої напрацювання є плагіатом та серйозним порушенням основних академічних стандартів.
2. Ви не повинні ділитися своїми розв’язками з іншими студентами, а також просити інших ділитися своїми розв’язками з Вами.
3. Якщо Ви отримували допомогу у вирішенні певного завдання, Ви маєте вказати це у звіті, а саме: від кого та яку допомогу отримали.

Практична 1: Основи Python

"The difference between stupidity and genius is that genius has its limits."
"Різниця між геніальністю і дурістю в тому, що у першої є свої межі."

– Альберт Ейнштейн

Вступ

Вирішуючи це завдання Ви реалізуєте кілька коротких функцій. Основна мета цього завдання – ознайомитись з Python, але як бонус, деякі функції будуть корисними для наступних завдань.

Опис завдання

Розв'язки слід вносити у `submission.py` між рядками:

```
# BEGIN_YOUR_CODE

# END_YOUR_CODE
```

Якщо необхідно, Ви можете додати інші допоміжні функції поза цим блоком. Не вносьте зміни до інших файлів, окрім `submission.py`. Не перейменовуйте також назви функцій у файлі `submission.py`, які Вам потрібно реалізувати.

Ваш код буде оцінено на двох типах тестів, **basic** та **hidden**, які Ви можете знайти у `grader.py`. Основні тести, які Вам повністю надаються, не випробують ваш код великими вхідними даними і не перевіряють хитромудрими граничними умовами.Приховані тести складніші і випробують ваш код. Вхідні дані прихованих тестів надаються в `grader.py`, але правильні результати - ні. Вони надаються вам для того, щоб переконатися, що функції не виходять з ладу та не потребують більше відведеного часу на виконання. Щоб запустити тести, вам потрібно буде, щоб файл `graderUtil.py` був у тому самому каталозі, що і ваш код та `grader.py`. Потім Ви можете запустити усі тести, набравши команду:

```
python3 grader.py
```

Це вкаже вам на те, чи пройшли ви основні тести і відобразить отримані бали. На прихованих тестах сценарій попередить, якщо ваш код потребує занадто багато часу для виконання або виходить з ладу.

Завдання для виконання

```
import collections
import math

############################################################
# Problem 1a

```

```
def findAlphabeticallyLastWord(text):
    """
    Given a string |text|, return the word in |text| that comes last
    alphabetically (that is, the word that would appear last in a dictionary).
    A word is defined by a maximal sequence of characters without whitespaces.
    You might find max() and list comprehensions handy here.
    """
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE

############################################################
# Problem 1b

def euclideanDistance(loc1, loc2):
    """
    Return the Euclidean distance between two locations, where the locations
    are pairs of numbers (e.g., (3, 5)).
    """
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE

############################################################
# Problem 1c

def mutateSentences(sentence):
    """
    Given a sentence (sequence of words), return a list of all "similar"
    sentences.
    We define a sentence to be similar to the original sentence if
      - it as the same number of words, and
      - each pair of adjacent words in the new sentence also occurs in the original sentence
        (the words within each pair should appear in the same order in the output sentence
         as they did in the orignal sentence.)
    Notes:
      - The order of the sentences you output doesn't matter.
      - You must not output duplicates.
      - Your generated sentence can use a word in the original sentence more than
        once.
    Example:
      - Input: 'the cat and the mouse'
      - Output: ['and the cat and the', 'the cat and the mouse', 'the cat and the cat', 'cat and the cat a
                (reordered versions of this list are allowed)
    """
    # BEGIN_YOUR_CODE (our solution is 20 lines of code, but don't worry if you deviate from this)
    raise Exception("Not implemented yet")

    # END_YOUR_CODE
```

```python

############################################################
# Problem 1d

def sparseVectorDotProduct(v1, v2):
    """
    Given two sparse vectors |v1| and |v2|, each represented as collections.defaultdict(float), return
    their dot product.
    You might find it useful to use sum() and a list comprehension.
    This function will be useful later for linear classifiers.
    """
    # BEGIN_YOUR_CODE (our solution is 4 lines of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE

############################################################
# Problem 3e

def incrementSparseVector(v1, scale, v2):
    """
    Given two sparse vectors |v1| and |v2|, perform v1 += scale * v2.
    This function will be useful later for linear classifiers.
    """

    # BEGIN_YOUR_CODE (our solution is 2 lines of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE

############################################################
# Problem 1f

def findSingletonWords(text):
    """
    Splits the string |text| by whitespace and returns the set of words that
    occur exactly once.
    You might find it useful to use collections.defaultdict(int).
    """
    # BEGIN_YOUR_CODE (our solution is 4 lines of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE

############################################################
# Problem 1g

def computeLongestPalindromeLength(text):
    """
    A palindrome is a string that is equal to its reverse (e.g., 'ana').
    Compute the length of the longest palindrome that can be obtained by deleting
```

```
    letters from |text|.
    For example: the longest palindrome in 'animal' is 'ama'.
    Your algorithm should run in O(len(text)^2) time.
    You should first define a recurrence before you start coding.
    """
    # BEGIN_YOUR_CODE (our solution is 19 lines of code, but don't worry if you deviate from this)

    raise Exception("Not implemented yet")
    # END_YOUR_CODE
```

Оцінювання

1. [3 бали] Реалізуйте функцію `findAlphabeticallyLastWord`
2. [1 бали] Реалізуйте функцію `euclideanDistance`
3. [1 бали] Реалізуйте функцію `mutateSentences`
4. [1 бали] Реалізуйте функцію `sparseVectorDotProduct`
5. [1 бали] Реалізуйте функцію `incrementSparseVector`
6. [1 бали] Реалізуйте функцію `findSingletonWords`
7. [2 бали] Реалізуйте функцію `computeLongestPalindromeLength`

Практична 2: Логістична регресія

"Грам власного досвіду коштує дорожче тонни чужих повчань."

– Магатма Ґанді

Вступ

Виконуючи це завдання, Ви познайомитесь з математичним апаратом, який лежить в основі навчання найпростішої нейронної мережі, що складається з одного нелінійного нейрона. Ця модель носить назву логістична регресія. Отриманий досвід буде корисним для подальшого розуміння принципу роботи глибинних нейронних мереж.

Класичне програмування та машинне навчання

`Комп'ютери` та `обчислення` допомагають нам досягати більш складних цілей і кращих результатів у вирішенні проблем, ніж ми могли б досягти самі. Однак, багато сучасних завдань вийшли за рамки обчислень через один основний обмежуючий фактор: **традиційно, комп'ютери можуть дотримуватися лише конкретних вказівок/інструкцій, які їм дають.**

Вирішення проблем з `програмування` вимагає написання конкретних покрокових інструкцій, які має виконувати комп'ютер. Ми називаємо ці кроки алгоритмами. У цьому випадку, комп'ютери можуть допомогти нам там, де ми:

1. Розуміємо як вирішити проблему.
2. Можемо описати проблему за допомогою чітких покрокових інструкцій, які комп'ютер може зрозуміти.

Методи машинного навчання дозволяють комп'ютерам "учитися" на прикладах. Вирішення проблем із застосуванням машинного навчання вимагає виявлення деякого шаблону[1], а потім, коли такий шаблон готовий, дозволяють, наприклад, нейронній мережі вивчити карту переходів між вхідними та вихідними даними. Ця особливість відкриває нові типи проблем, де комп'ютери можуть допомогти нам у їх розв'язанні, за умови, коли ми:

1. Визначили шаблон проблеми.
2. Маємо достатньо даних, що ілюструють шаблон.

На рисунку 1 графічно показана відмінність класичного програмування від машинного навчання.

[1] Пошук прикладів, що висвітлюють обидві сторони шаблону: вхід і вихід.

Рис. 1: Відмінність класичного програмування від машинного навчання.

Письмове завдання

Покажіть, що похідна сигмоїди дорівнює цьому виразу:

$$\frac{d\hat{y}}{dz} = \frac{d\sigma(z)}{dz} = \sigma(z)\big(1 - \sigma(z)\big), \tag{1}$$

де $\hat{y} = \sigma(z) = \frac{1}{1+\exp(-z)}$

Завдання з програмування

Відкрийте завдання: https://github.com/YKochura/ai-lab/blob/main/lab3/logistic_regression.ipynb

Вам потрібно імплементувати декілька функцій для прямого та зворотного поширення одного навчального прикладу логістичної регресії. Функції, які потрібно імплементувати позначено у завданні так:

```
# TODO
```

Розміщуйте свою реалізацію функцій між

```
# BEGIN_YOUR_CODE

# END_YOUR_CODE
```

Коли усі функції будуть реалізовані, дослідіть два пункти, які подані у кінці завдання:

1. Повторіть кроки 1-5 та з'ясуйте як зміниться значення цільової функції, ваг та зсуву моделі. Швидкість навчання використовуйте $alpha = 0.0001$
2. Повторіть кроки 0-5 для більшої швидкості навчання $alpha = 0.003$. Порівняйте отримані результати для $alpha = 0.0001$

Оцінювання

Ваша оцінка за виконання завдання буде залежати від:

- 10% – письмове завдання
- 60% – завдання з програмування
- 30% – підготовлено звіт у якому подано розв'язок письмового завдання та досліджено зміну цільової функції, ваг та зсуву моделі залежно від швидкості навчання та кількості ітерацій навчання. Очікується формальний звіт, написаний в LaTeX.

  Шаблон за яким потрібно підготувати звіт поданий нижче на наступних сторінках:

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Системи штучного інтелекту

Логістична регресія

Практична робота #2

Виконав(ла):
Ім’я Прізвище
Група:
ІО-ххх
Курс:
X

2 вересня 2022 р.

1 Логістична регресія

1.1 Письмове завдання

Покажіть, що похідна сигмоїди дорівнює цьому виразу:

$$\frac{d\hat{y}}{dz} = \frac{d\sigma(z)}{dz} = \sigma(z)\big(1 - \sigma(z)\big), \tag{1}$$

де $\hat{y} = \sigma(z) = \frac{1}{1+\exp(-z)}$

Розв'язок:

1.2 Завдання з програмування

Подайте у звіті імплементовані Вами функцій. Це можна зробити через зовнішній файл, наприклад так:

Лістинг 1: Перевірка на парність введеного числа.

```python
# Python program to check if the input number is odd or even.
# A number is even if division by 2 gives a remainder of 0.
# If the remainder is 1, it is an odd number.

num = int(input("Enter a number: "))
if (num % 2) == 0:
   print("{0} is Even".format(num))
else:
   print("{0} is Odd".format(num))
```

Або вставивши безпосередньо потрібні рядки коду в оточення `lstlisting`:

Лістинг 2: Перетин двох масивів.

```python
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```

Без рамки та без підпису:

```python
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```

1.3 Результати експериментів

Тут Ви повинні подати детальну інформацію про результати свого дослідження стосовно зміни значень цільової функції, ваг та зсуву моделі залежно від швидкості навчання та кількості ітерацій навчання для одного навчального прикладу (додаткові деталі знаходяться у кінці `logistic_regression.ipynb`).

Для представлення результатів використовуйте таблиці, графіки, рисунки. Приклад таблиці 1. Приклад рисунку 1.

Табл. 1: Приклад таблиці.

Ім'я оператора		Синтаксис
Присвоєння		a = b
Додавання		a + b
Віднімання		a - b
Унарний плюс		+a
Унарний мінус		-a
Множення		a * b
Ділення		a / b
Залишок від ділення		a % b
Інкремент	префікс суфікс	++a a++
Декремент	префікс суфікс	- -a a- -

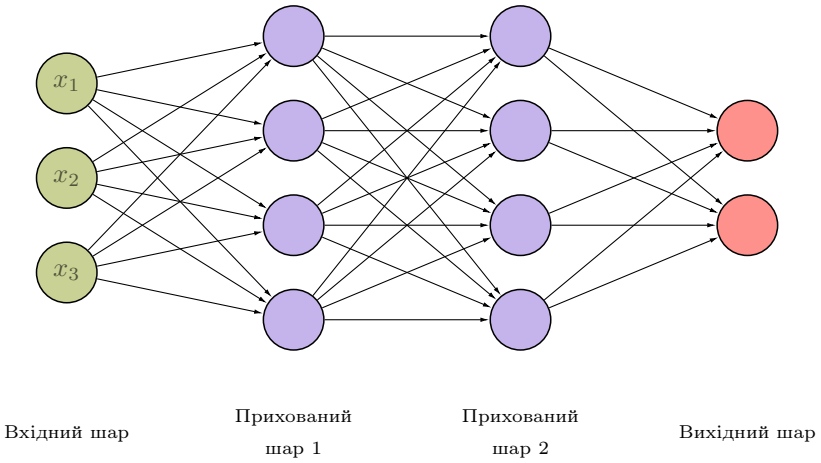


Рис. 1: Приклад представлення нейронної мережі.

1.4 Допомога

У цьому розділі потрібно залишити додаткові деталі про виконання цієї роботи. Тобто, якщо Ви обговорювали завдання або працювали над ним у групі – потрібно це зазначати: з ким працювали та що кожен учасник виконував. Якщо отримували допомогу – вказати від кого та яку допомогу було отримано. Якщо використовували додаткові матеріали (блокноти Kaggle, github тощо) – залишите посилання на джерела. Якщо завдання було виконано особисто Вами і Ви не отримували ніякої допомоги – так і напишіть. Використання будь-яких чужих матеріалів та представлення їх за свої напрацювання є плагіатом та серйозним порушенням основних академічних стандартів.

Приклад посилань на два джерела одночасно [1, 2].

1.5 Висновки

Резюмуйте пророблену Вами роботу, відобразіть свої спостереження щодо змін цільової функції, ваг та зсуву моделі залежно від швидкості навчання та кількості ітерацій навчання.

Література

[1] M. Mathur. Cnn using keras(100% accuracy). Kaggle. [Online]. Available: https://www.kaggle.com/madz2000/cnn-using-keras-100-accuracy

[2] R. Jain. Deep learning using sign langugage. Kaggle. [Online]. Available: https://www.kaggle.com/ranjeetjain3/deep-learning-using-sign-langugage

Практична 3: Початок роботи з глибинним навчанням

"Той, хто вчиться, але не мислить – втратить себе.
Той, хто мислить, але не вчиться – занапастить себе."

– Конфуцій (551 - 479 до н. е.)

Для того, щоб успішно завершити виконання практичного завдання, Вам знадобиться:

1. Знання англійської: можливість розуміти основний зміст поставленого завдання.
2. Базові знання **Python**: цикли, умови, функції та змінні.

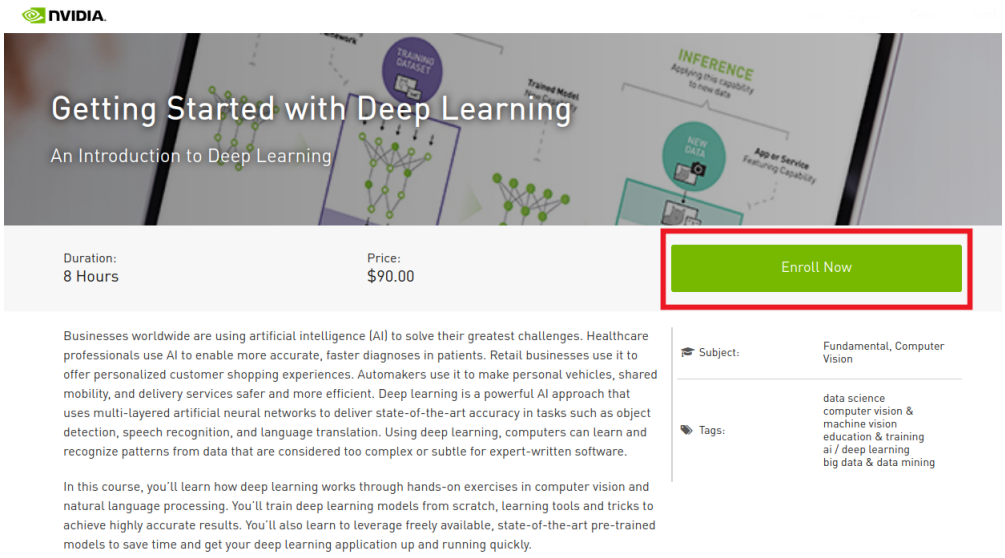
Обліковий запис

Для отримання БЕЗКОШТОВНОГО доступу до матеріалів NVIDIA Deep Learning Institute, Вам потрібно **створити** або **використати уже існуючий** екаунт NVIDIA. Для цього перейдіть за цим посиланням: courses.nvidia.com/join.

Практичне завдання #2

Перейдіть за посиланням, яке указане нижче та виберіть **Enroll Now**:

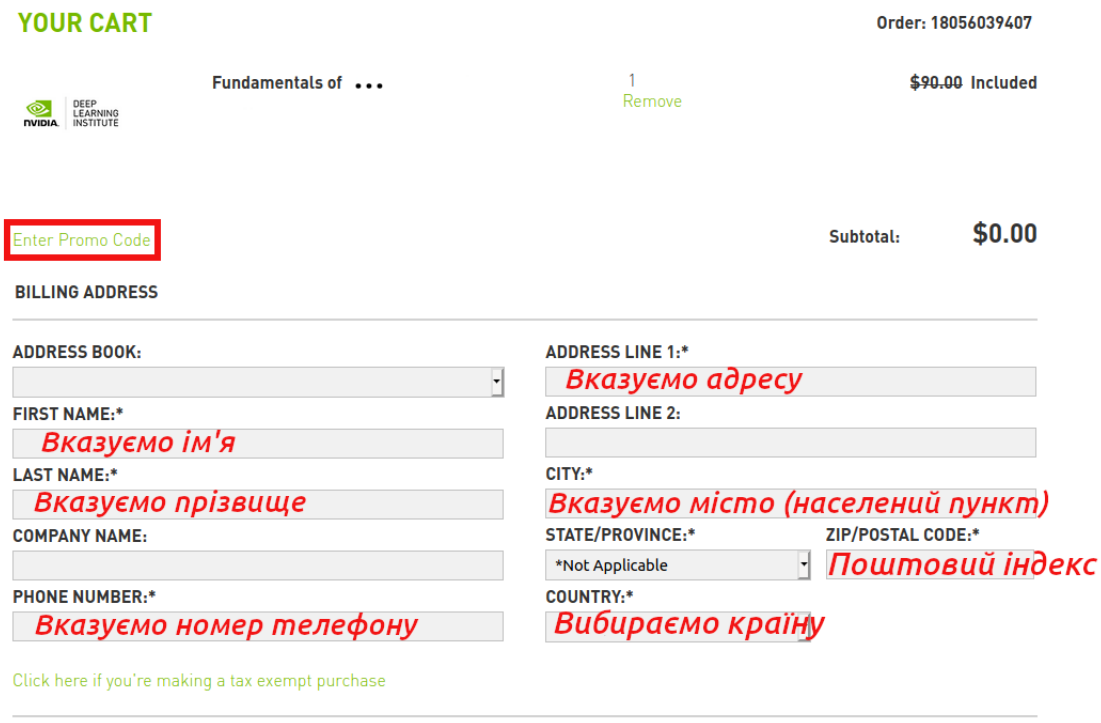
Getting Started with Deep Learning



1. У полі **Enter Promo Code** потрібно вказати (уточніть у викладача):

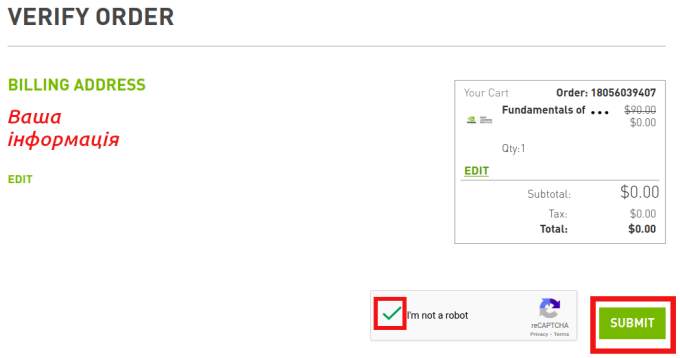
   DLITEACHxxxx_xx_xxx_xx

2. Заповніть також поля, які позначені червоним кольором на цьому зображенні:

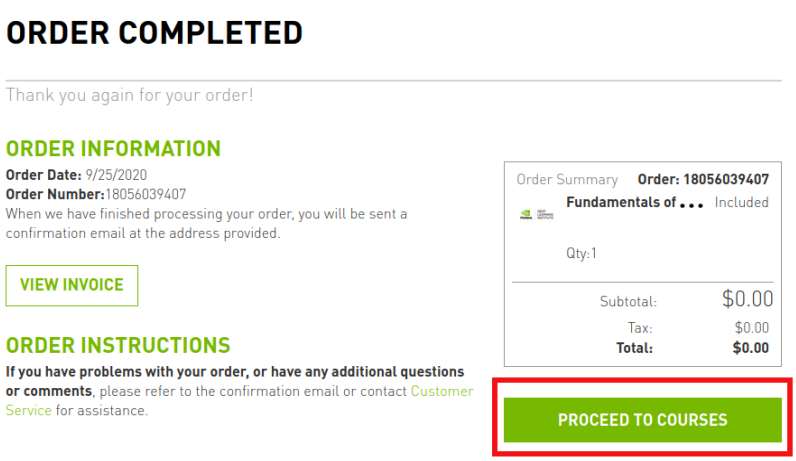


3. Для продовження потрібно натиснути **CONTINUE** (Ви перейдете до останнього етапу):

4. Перевіряємо інформацію про себе, підтверджуємо, що не є роботом та натискаємо для завершення **SUBMIT**:



5. Для переходу до матеріалів курсу натискаємо на **PROCEED TO COURSES**:



Що ви вивчите після виконання цього завдання?

На момент завершення виконання цього завдання Ви зможете:

1. Застосовувати методи глибинного навчання для задач класифікації зображень та виявлення об'єктів.
2. Експериментувати з даними, гіперпараметрами, структурою мережі та іншими стратегіями, які дозволяють підвищити продуктивність Вашої моделі.
3. Застосовувати техніку передавального навчання (transfer learning) між моделями для досягнення більшої продуктивності з використанням меншої кількістю навчальних даних та обчислень.

Практична 4: Розпізнавання рукописних цифр

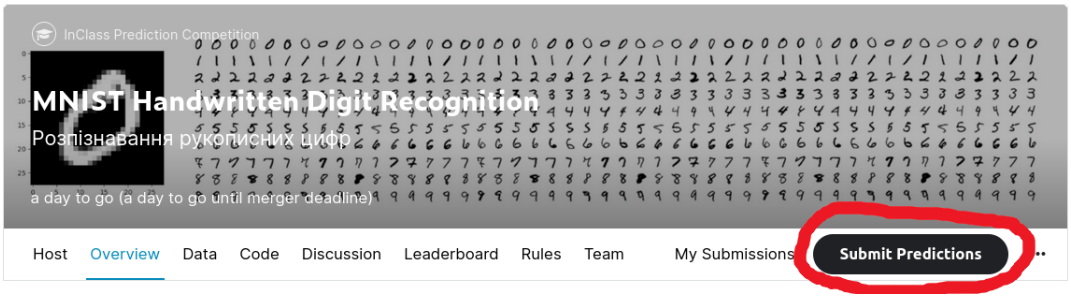
"Small daily improvements over time create stunning results."
"Маленькі щоденні покращення – ключ до приголомшливих довгострокових результатів."

– Робін Шарма

Опис завдання

Розпізнайте рукописні цифри з MNIST in CSV датасету за допомогою нейронних мереж.

1. Для виконання цього завдання Вам знадобиться аккаунт kaggle (створіть або використовуйте уже існуючий).
2. Перейдіть за наступним посиланням, ознайомтесь та прийміть умови змагання: https://www.kaggle.com/t/5b6195126f5542d282a30d346a73666e
3. Ваше завдання – натренувати модель так, щоб вона розпізнавала якнайкраще рукописні цифри з тестового датасету змагання (`test.csv`).
4. Отриманий Вами прогноз за моделлю слід сабмітити на сайт змагання: вкладка `"Submit Predictions"`.



5. Деякі приклади як це робити можна знайти `[тут]`, `[тут]` або `[тут]`.

Оцінювання

Ваша оцінка за виконання завдання буде залежати від:

- 40% - продуктивність на Kaggle. Максимальний бал, якщо точність прогнозу на тестовому наборі > 0.995
- 10% - детально прокоментований скрипт отриманої моделі: `Прізвище Ім'я_група.ipynb`
- 50% - звіт. У звіті подаєте детальний опис набору даних, описуєте процес налаштування моделі та підготовку даних до навчання, результати експериментів, висновки. Очікується досить формальний звіт, написаний в LaTeX.

  Шаблон за яким потрібно підготувати звіт поданий нижче на наступних сторінках:

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Системи штучного інтелекту

MNIST: розпізнавання рукописних цифр

Практична робота #4

Виконав(ла):
Ім’я Прізвище
Група:
ІО-ххх
Курс:
X

15 листопада 2021 р.

1 MNIST: розпізнавання рукописних цифр

1.1 Набір даних

Опишіть свій набір даних: скільки прикладів було використано для навчання (training) / перевірки (validation) / тестування (test)? Чи виконували Ви попередню обробку (preprocessing), нормалізацію (normalization) або збільшення даних (data augmentation)? Якщо робили – коротко опишіть як це було зроблено. Яка роздільна здатність ваших зображень? Подайте кілька прикладів з набору у вигляді зображень. Також Вам слід вказати вхідні фічі (ознаки, характеристики), які використовувались для навчання моделі. Фіча – це та характеристика, від якої безпосередньо залежить вихідний результат передбачення. Наприклад, для визначення діабету у людини, такими фічами будуть: вік, стать, індекс маси тощо. Залиште посилання на використаний набір даних. Наприклад, так: Для виконання цього завдання було використано набір даних MNIST in CSV [1].

1.2 Структура моделі

Подаєте детальний опис архітектури нейронної мережі (скільки шарів, якого розміру тощо). Якщо Ви хочете супроводжувати свій опис рядками коду – можете це зробити через зовнішній файл, наприклад так:

Лістинг 1: Перевірка на парність введеного числа.

```
# Python program to check if the input number is odd or even.
# A number is even if division by 2 gives a remainder of 0.
# If the remainder is 1, it is an odd number.

num = int(input("Enter a number: "))
if (num % 2) == 0:
   print("{0} is Even".format(num))
else:
   print("{0} is Odd".format(num))
```

Або вставивши безпосередньо потрібні рядки коду в оточення `lstlisting`:

Лістинг 2: Перетин двох масивів.

```
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```

Без рамки та без підпису:

```
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```

1.3 Результати експериментів

Тут Ви повинні подати детальну інформацію про те, якими гіперпараметрами управляли (наприклад, чому використовували швидкість навчання (learning rate) X для градієнтного спуску, який був ваш розмір mini-batch?) та як Ви їх обирали. Чи робили Ви перехресну перевірку (cross-validation), якщо так, то скількох кратну? Обов'язково вкажіть і поясніть, якими є ваші основні метрики: accuracy, precision, AUC тощо. Надайте рівняння для метрик.

Приклад подання рівняння. На практиці як правило сигмоїду використовують на вихідному шарі нейронної мережі, отримане значення можна також інтерпретувати як ймовірність:

$$P(y = 1|z) = \sigma(z) = \frac{1}{1 + e^{-z}}, \tag{1}$$

де y – очікуване значення моделі.

Для представлення результатів використовуйте таблиці та графіки. Отримані результати, які відображені Вами у таблицях або графіках, супроводжуйте описом у тексті звіту. Приклад таблиці 1. Приклад рисунку 1.

Табл. 1: Приклад таблиці.

Ім'я оператора		Синтаксис
Присвоєння		a = b
Додавання		a + b
Віднімання		a - b
Унарний плюс		+a
Унарний мінус		-a
Множення		a * b
Ділення		a / b
Залишок від ділення		a % b
Інкремент	префікс суфікс	++a a++
Декремент	префікс суфікс	- -a a- -

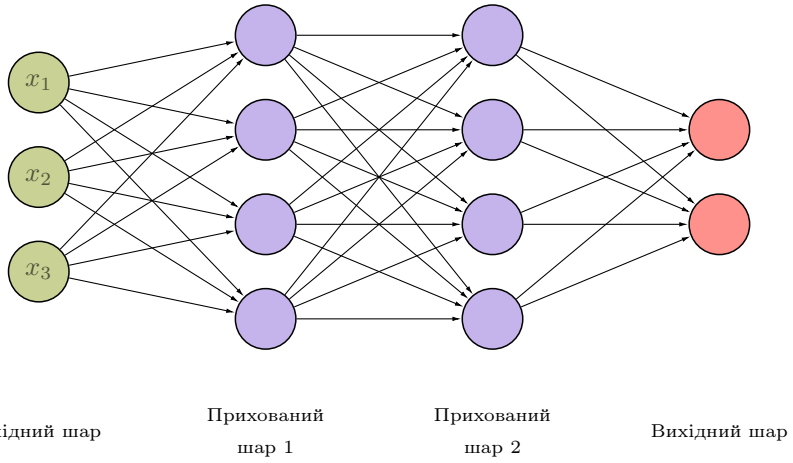


Рис. 1: Приклад представлення нейронної мережі.

1.4 Допомога

У цьому розділі потрібно залишити додаткові деталі про виконання цієї роботи. Тобто, якщо Ви обговорювали завдання або працювали над ним у групі – потрібно це зазначати: з ким працювали та що кожен учасник виконував. Якщо отримували допомогу – вказати від кого та яку допомогу було отримано. Якщо використовували додаткові матеріали (блокноти Kaggle, github тощо) – залишите посилання на джерела. Якщо завдання було виконано особисто Вами і Ви не отримували ніякої допомоги – так і напишіть. Використання будь-яких чужих матеріалів та представлення їх за свої напрацювання є плагіатом та серйозним порушенням основних академічних стандартів.

Приклад посилань на два джерела одночасно [2, 3].

1.5 Висновки

Резюмуйте пророблену Вами роботу, відобразіть свої спостереження щодо налаштування мережі. Чому була обрана саме така архітектура нейронної мережі, яка представлена у роботі? Для яких гіперпараметрів модель працювала краще за все?

Література

[1] MNIST in CSV. Kaggle. [Online]. Available: https://www.kaggle.com/oddrationale/mnist-in-csv/code

[2] M. Mathur. Cnn using keras(100% accuracy). Kaggle. [Online]. Available: https://www.kaggle.com/madz2000/cnn-using-keras-100-accuracy

[3] R. Jain. Deep learning using sign langugage. Kaggle. [Online]. Available: https://www.kaggle.com/ranjeetjain3/deep-learning-using-sign-langugage

Проєкт: інструкції для виконання

"Education is only a ladder to gather fruit from the tree of knowledge, not the fruit itself."
"Освіта – це лише драбина для збору фруктів з дерева знань, а не сам фрукт."

– Альберт Ейнштейн

Опис завдання

Над проєктною роботою можна працювати у групах (до двох людей) або самостійно. **Мета проєктної роботи** – застосовувати та розглянути передові прийоми та розробки в галузі штучного інтелекту, які дотичні до цього курсу та викликають у Вас зацікавленість.

Ваше перше завдання – вибрати тему проєкту. Ви можете вільно обрати тему самі, проте якщо складно визначитися з темою, будь ласка, повідомте мене і я запропоную для Вас кілька ідей.

Виконати можна один із трьох видів проєктів:

1. **Проєкт застосунку.** Виберіть напрямок, який Вас цікавить (наприклад, *медицина, мова, торгівля, енергетика, спорт, ігри, робототехніка тощо*) і дослідіть на практиці провідні напрацювання за обраним напрямком, що опубліковані у статтях, на github тощо.
2. **Алгоритмічний проєкт.** Виберіть проблему (задачу) або сімейство проблем і розробіть новий алгоритм навчання або новий варіант існуючого алгоритму для її вирішення.
3. **Теоретичний проєкт.** Доведіть деякі цікаві/нетривіальні властивості існуючого або нового алгоритму навчання.

Проєкти над якими Ви будете працювати можуть поєднувати елементи застосунку, алгоритмів та теорії.

Щоб визначитися із темою, Ви можете обирати або область застосування, яка Вас цікавить, або якийсь напрямок штучного інтелекту у якому хочете краще розібратися та вивчити. Отже, виберіть щось, чим зможете захопитись і приступайте до роботи. Для натхнення Ви можете переглянути деякі останні дослідження з машинного навчання. Дві основні конференції машинного навчання – це ICML та NeurIPS. Ви можете знайти статті з нещодавніх конференцій за цими гіперпосиланнями: ICML та NeurIPS.

Обравши цікаву тему, перегляньте існуючі дослідження з відповідної теми, здійснивши пошук ключових слів в академічній пошуковій системі, наприклад: http://scholar.google.com.

Іншим важливим моментом роботи над проєктом є визначення одного або декількох наборів даних, що підходять для вашої теми. Якщо ці дані потребують значної попередньої обробки відповідно до вашого завдання або Ви збираєтеся самостійно зібрати необхідні дані, майте на увазі, що це лише одна частина очікуваної проєктної роботи, яка може зайняти чимало вашого часу.

Відтворення результатів, які представлені у статті, може бути хорошим способом навчання. Однак, замість того, щоб просто тиражувати статтю, шляхом відтворення результатів, спробуйте використати розглянутий у статті підхід в іншому застосуванні та порівняти результати або зробити якийсь аналіз того, як кожен компонент моделі впливає на кінцеву продуктивність.

Ваша проєктна робота повинна складатися з наступних частин:

* **Анотація**
* **Вступ**
* **Літературний огляд**
* **Набір даних та фічі**
* **Методи**
* **Експерименти & Результати**
* **Висновки**
* **Внесок**
* **Література**
* **Лістинг коду**

Кінцевий звіт (без двох останніх розділів: **Література** та **Лістинг коду**) повинен бути не більше 6 сторінок (включаючи усі рисунки і таблиці).

Набори даних

Ви можете вільно користуватися наявними наборами даних для свого проєкту, що знаходяться у відкритому доступі або зібрати та створити власний.

* Kaggle – тут можна знайти деякі набори даних.
* Деякі набори даних для NLP
* OpenAI Gym – середовище для навчання з підкріпленням (reinforcement learning).

Обчислювальні ресурси

Для виконання проєктної роботи Ви можете використовувати безкоштовно обчислювальні ресурси, які пропонують:

* Google Colab
* Kaggle

Оцінювання

Під час оцінювання проєктної роботи буде братися до уваги:

* **Постановка завдання**. Чітко визначено завдання та зрозуміла мотивація для вирішення обраної проблеми. Новизна проблеми, технічна якість вирішення проблеми та значущість роботи. Виконано літературний огляд.
* **Дані та експерименти**. Чітко описали свій набір даних або середовище навчання та експерименти, які провели. Перерахували свої результати та метрики.
* **Методи**. Детально описано використані у роботі алгоритми навчання.

* **Висновок**. Прозора інтерпретація отриманих результатів.

Максимальна оцінка за виконання цього завдання становить 30 балів.

Звіт проєкту

Очікується досить формальний звіт, який буде підготовлено в LaTeX. Шаблон за яким потрібно підготувати звіт проєкту подано нижче на наступних сторінках:

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Системи штучного інтелекту

Проєктна робота

Назва проєкту

Виконали

Ім’я Прізвище
Група: IO-xxx
Курс: X

Ім’я2 Прізвище2
Група: IO-xxx
Курс: X

29 листопада 2021 р.

1 Анотація [≈ 1 абзац]

Анотація повинна бути розміром в один абзац, містити в собі мотивацію обраної теми та пояснення застосованого методу/методів і отриманих результатів.

2 Вступ [≈ 0.5 сторінки]

Поясніть проблему та чому вона важлива. За необхідності дайте трохи теорії. Чітко вкажіть, які вхідні та вихідні дані використовували. Наприклад: "Вхідними даними нашого алгоритму є {зображення, вік пацієнта, кількості опадів, відео у відтінках сірого тощо}. Потім ми використали {SVM, нейронну мережу, лінійну регресію тощо}, щоб вивести прогнозований {вік, ціну акцій, тип раку, музичний жанр тощо}." Це дуже важливо, оскільки різні команди, як правило, мають різні дані на вході/виході, що охоплюють різні способи застосувань. Якщо явно про це говорити – це полегшить читачу зрозуміти з чим Ви працювали.

1. Один
2. Два
3. Три
4. ...

- Один
- Два
- Три
- ...

3 Літературний огляд [≈ 0.5 сторінки]

Вам слід знайти існуючі статті, згрупувати їх за категоріями на основі їх підходів та обговорити їх сильні та слабкі сторони, а також те, чим вони схожі та відрізняються від вашої роботи. Google Scholar може допомогти з пошуком: http://scholar.google.com (Ви можете натиснути кнопку **"Cite** і це генерує посилання на матеріал у необхідному для вас форматі: MLA, APA, BibTeX тощо). Приклад посилання на джерела [1, 2].

4 Набір даних та фічі [≈ 0.5 сторінки]

Опишіть свій набір даних: скільки прикладів було використано для навчання (training)/перевірки (validation)/тестування (test)? Чи виконували Ви попередню обробку (preprocessing), нормалізацію (normalization) або збільшення даних (data augmentation)? Якщо навчальними даними у Вас були зображення, вкажіть яка їх роздільна здатність? Додайте посилання на те, звідки ви брали дані для вирішення обраної задачі. Подайте на рисунку кілька прикладів із набору даних. Також Вам слід вказати фічі (ознаки), якими користувались. Фіча – це та ознака, від якої безпосередньо залежить вихідний результат передбачення моделі. Наприклад, для визначення діабету у людини, такими фічами будуть: вік, стать, індекс маси тощо. Якщо витягували фічі за допомогою перетворень Фур'є, word2vec, гістограми орієнтованих градієнтів (HOG), PCA (Principal component analysis), ICA (Independent component analysis) тощо, обов'язково про це вкажіть. Приклад рисунку 1.

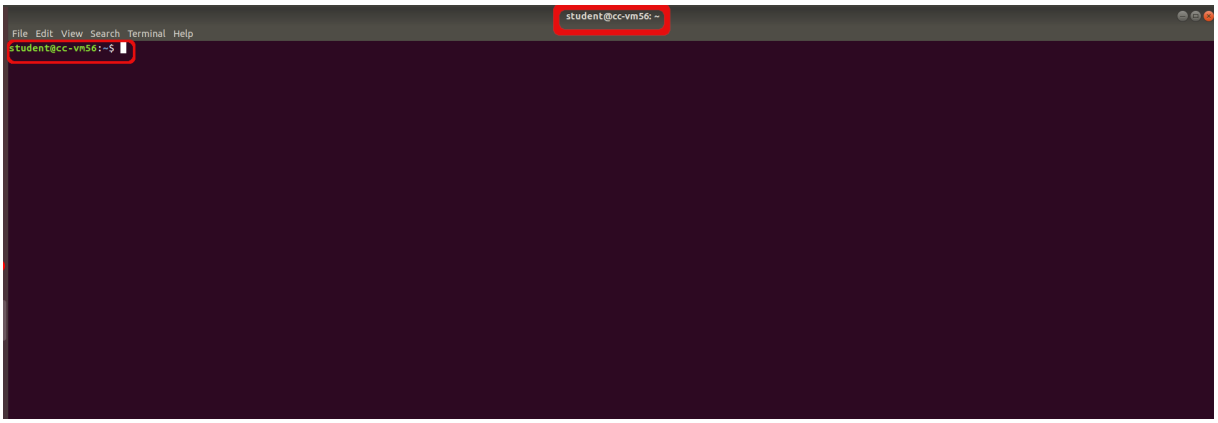


Рис. 1: Опис рисунку.

5 Методи [≈ 1-1.5 сторінки]

Опишіть використані у роботі алгоритми навчання. Обов'язково включіть відповідні математичні примітки, формули. Наприклад, Ви можете коротко включити мету, а також формулу оптимізації SVM або вказати активаційну функцію/функції, які використовувались у прихованих шарах та на виході. Для кожного апробованого алгоритму дайте короткий опис (≈ 1 абзац). Ви маєте показати своє розуміння того, як працюють ці алгоритми навчання.

6 Експерименти & Результати [≈ 1-3 сторінки]

Тут Ви маєте надати детальну інформацію про те, якими гіперпараметрами управляли (наприклад, чому використовували швидкість навчання (learning rate) X для градієнтного спуску, який був ваш розмір mini-batch?) та як Ви їх обирали. Чи робили Ви перехресну перевірку (cross-validation), якщо так, то скількох кратну? У k-кратній перехресній перевірці, вихідна вибірка випадковим чином розподіляється на k рівних підмножин. Перш ніж перераховувати свої результати, обов'язково вкажіть і поясніть, якими є ваші основні метрики: accuracy, precision, AUC тощо. За необхідності надайте рівняння для метрик. Для представлення результатів використовуйте таблиці та графіки з коротким поясненням та аналізом. Приклад таблиці 1.

Табл. 1: Арифметичні оператори.

Ім'я оператора		Синтаксис
Присвоєння		a = b
Додавання		a + b
Віднімання		a - b
Унарний плюс		+a
Унарний мінус		-a
Множення		a * b
Ділення		a / b
Залишок від ділення		a % b
Інкремент	префікс суфікс	++a a++
Декремент	префікс суфікс	- -a a- -

7 Висновки [≈ 1-2 абзаци]

Резюмуйте пророблену Вами роботу та повторіть ключові моменти отриманих результатів. Яку проблему вирішували та які алгоритми були найефективнішими для цього? Чому, на вашу думку, деякі алгоритми працювали краще за інші? Якби у Вас було більше часу, більше членів команди або більше обчислювальних ресурсів, що б Ви дослідили, спробували?

8 Внесок [≈ 1 абзац]

Цей розділ повинен містити інформацію про те, над чим кожен член команди працював та який зробив внесок у проєкт.

Література

[1] T. I. A. for Research on Cancer. (2018) Latest global cancer data: Cancer burden rises to 18.1 million new cases and 9.6 million cancer deaths in 2018. [Online]. Available: https://www.who.int/cancer/PRGlobocanFinal.pdf

[2] S. Anisimov, D. Pandelidis, and V. Maisotsenko, "Numerical study of heat and mass transfer process in the maisotsenko cycle for indirect evaporative air cooling," *Heat Transfer Engineering*, pp. 1–40, January 28 2016. [Online]. Available: http://dx.doi.org/10.1080/01457632.2016.1142314

Лістинг коду

Розмістіть тут програмну реалізацію цього проєкту.

Лістинг коду через зовнішній файл, наприклад так (`hello.c` та `test.py` знаходяться у директорії `code` шаблону LaTeX):

Лістинг 1: Назва лістингу.

```c
#include <stdio.h>

// main prints "hello world" to standard output.
int main(int argc, char **argv) {
  printf("hello world\n");
  return 0;
}
```

Лістинг 2: Перевірка на парність введеного числа.

```python
# Python program to check if the input number is odd or even.
# A number is even if division by 2 gives a remainder of 0.
# If the remainder is 1, it is an odd number.

num = int(input("Enter a number: "))
if (num % 2) == 0:
   print("{0} is Even".format(num))
else:
   print("{0} is Odd".format(num))
```

Або вставте безпосередньо потрібні рядки коду в оточення `lstlisting` так як показано нижче:

Лістинг 3: Перетин двох масивів.

```
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```

Без рамки та без підпису:

```
def intersection(array_1, array_2):
  element_1 = set()
  intersected = []
  for i in range(0, len(array_1)):
    element_1.add(array_1[i])
  already_added = set()
  for j in range(0, len(array_2)):
    if array_2[j] in element_1 and array_2[j] not in already_added:
      intersected.append(array_2[j])
      # MISSING LINE HERE.
      already_added.add(array_2[j])
  return intersected
```