Лабораторна робота №1. Основи роботи з PyTorch

Мета: засвоїти основні відомості про роботу з фреймворком PyTorch. Навчитись розпізнавати не типові елементи в даних.

Теоретичні відомості

Основні відомості про PyTorch.

1. Інсталяція:

- Для більшості систем з встановленим python3 достатньо виконати в терміналі команду:

```
pip install torch===1.4.0 torchvision===0.5.0 -f
https://download.pytorch.org/whl/torch_stable.html
```

- Якщо процесудра вище не працює по якійсь причині, потрібно виконати детальні інструкції які доступні за посиланням
https://pytorch.org/get-started/locally/

Поняття Тензора

Те́нзор (від лат. *tendere*, «тягнутись, простиратися») — математичний об'єкт, що узагальнює такі поняття як скаляр, вектор, ковектор, лінійний оператор і білінійна форма. Вивченням тензорів займається тензорне числення.

Тензор - ключовий елемент в структурі фреймворка PyTorch. Усі операції в межах PyTorch виконуються з тензорами. Нижче наведені декілька прикладів операцій з поясненнями

- Створення тензорів, приклади:

  Створення тензорів розміром в 1 елемент

```
number1 = torch.Tensor(1)
number2 = torch.randn(1)
number3 = torch.ones(1)
number4 = torch.zeros(1)
print("{}, {}, {}, {}".format(number1, number2, number3, number4))
```

  Output: tensor([3.1389e-43]), tensor([-0.5335]), tensor([1.]), tensor([0.])

  Створення матиць розміром 2x2:

```
number1 = torch.Tensor((2,2))
number2 = torch.randn((2,2))
number3 = torch.ones((2,2))
number4 = torch.zeros((2,2))
print("{}, {}, {}, {}".format(number1, number2, number3, number4))
```

  Output: tensor([2., 2.]), tensor([[-0.8699, 0.1639], [-0.0022, -0.6090]]),
  tensor([[1., 1.], [1., 1.]]), tensor([[0., 0.], [0., 0.]])

  Створення тензорів з існуючого python списку

```
number1 = torch.tensor([[1., -1.], [1., -1.]])
print("{}".format(number1))
```

  Output: tensor([[1., -1.], [1., -1.]])

  Базові операції:

```
x = torch.tensor([[1., -1.], [1., 1.]])
out = x.pow(2).sum()
```

Детальний перелік операцій доступний за посиланням
https://pytorch.org/docs/stable/tensors.html

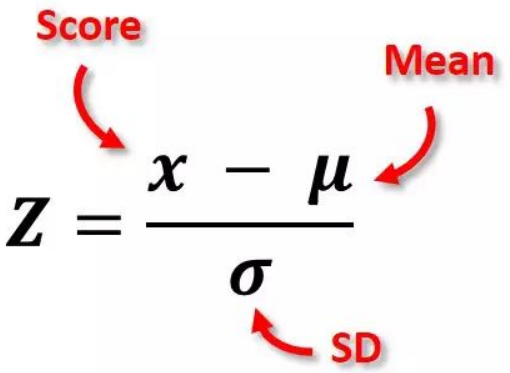
Додаткові посилання:

1. https://pytorch.org/tutorials/
2. https://pytorch.org/docs/stable/index.html

Визначення нетипових елементів. Z-score

Визначення нетипових елементів(outliers detection) - процедура визначення елементів в датасеті, які сильно відрізняються від решти елементів цього датасету. Це частий крок який використовують при обробці та аналізі даних.

Один за найпростіших алгоритмів визначення - Z-score. Z-score показує наскільки сильно відрізняється елемент даних від типового(середнього). Усі елементи які мають Z-score > 3 вважаються нетиповими.



Завдання: Порахувати outliers для датасету і колонок згідно варіантів, а саме:

1. Порахувати z-score незалежно для кожної з згаданих цифрових колонок. Для обрахунку використати лише PyTorch.
2. Агрегувати пораховані z-score(наприклад усередненням). Візуалізувати датасет і знайдені нетипові дані(ті, для яких z-score > 3) на scatter plot. Нетипові дані і решту датасету звізуалізувати різними кольорами.
3. Якщо серед згаданих колонок є категоріальні дані, порахувати z-score для кожної з колонок в межах кожної з можливих категорій. Наприклад для iris датасету колонка ‘species’ може набувати значень 'setosa', 'versicolor', 'virginica'. Потрібно порахувати z-score для решти колонок лише для значень які в колонці ‘species’ мають значення ‘setosa’, після цього повторити те саме для значень які мають ‘versicolor’ і завершити обрахунком z-score для значень що мають ‘virginica’ в полі ‘species’
4. Аналогічно до пункту 2, агрегувати отримані значення і визначити нетипові дані.

Варіанти

Варіант #	Датасет	Назви колонок з якими потрібно зробити аналіз
1	titanic	'survived',’age', 'fare'
2	iris	'sepal_length', 'sepal_width', 'species'
3	planets	'method', 'number', 'orbital_period'
4	titanic	'pclass', 'sibsp', 'parch'
5	iris	'petal_length', 'petal_width', 'species'
6	planets	'mass', 'distance', 'year'
7	tips	'total_bill', 'tip', 'sex', 'day'
8	diamonds	'carat', 'cut',, 'clarity',
9	tips	'total_bill', 'smoker', 'time', 'size'
10	diamonds	'depth', 'table', 'price','color',
11	exercise	'diet', 'pulse', 'time', 'kind'
12	diamonds	'carat', 'x', 'y', 'z'

Приклад реалізації на numpy:

```
#%%
import seaborn as sns
import numpy as np
import pandas as pd
#%%
tit = sns.load_dataset("titanic")
iris = sns.load_dataset("iris")
diamonds = sns.load_dataset('diamonds')
tips = sns.load_dataset('tips')
planets = sns.load_dataset('planets')


#%%
```

```
class Zscore:

    def __init__(self, np_columns: np.ndarray):
        self.mean = np.mean(np_columns)
        self.sigma = np.std(np_columns, axis=0)

    def get_score(self, x):
        return (x - self.mean)/self.sigma

    def get_avarage_score(self, x):
        return np.mean((x - self.mean)/self.sigma)

#%%
score_calc = Zscore(tit["age"])
#%%
scores = score_calc.get_score(tit["age"])
#%%
```