Практикум 1. Розв'язання задач регресії засобами TensorFlow Python

Недашківська Н.І.

1 Хід виконання роботи

Написати програму, яка задає і навчає регресійну модель методами градієнтного спуску:

1. Завантажити початкові дані.
2. Представити дані графічно.
3. Ініціалізувати параметри: швидкість навчання, кількість епох (=100) і додатково, якщо задано згідно з варіантом, кількість пакетів і/або параметр регуляризації.
4. Ініціалізувати вектор параметрів.
5. Реалізувати функцію, яка задає модель, наприклад:
   - лінійну регресію,
   - поліноміальну регресію,
   - іншу криву, яку підібрати відповідно до заданих даних.
6. Задати функцію втрат (одну згідно з варіантом):
   - MSE,
   - MSE з регуляризацією за нормами L_1 або L_2.
7. Задати операцію, яка буде викликатися на кожній ітерації алгоритма навчання:
   - алгоритм градієнтного спуску,
   - алгоритм міні-пакетного градієнтного спуску,
   - алгоритм градієнтного спуску з моментом,
   - алгоритм Adagrad,
   - алгоритм Adadelta,
   - алгоритм Adam.

8. Виконати навчання моделі.

9. Виводити значення функції втрат через кожні 10 епох.

10. Використовуючи Saver, зберегти контрольні точки через регулярні інтервали під час навчання. В кінці навчання зберегти результуючу модель. Відновити останню контрольну точку при запуску, якщо навчання було перервано.

11. Налаштувати гіперпараметр швидкість навчання і додатково, якщо задано згідно варіанту, розмір міні-пакета. Подивитися на форму кривої навчання.

12. Дослідити різні значення параметра регуляризації і підібрати найкраще з них, якщо згідно з варіантом задано регуляризовану функцію втрат. В цьому випадку дані мають бути попередньо розбиті на навчальний та перевірочний набори.

13. Побудувати графік з початковими даними та лінією регресії.

2 Варіанти завдань для групи КА-87

Номери варіантів вказано у файлі Group3... xls .

1. Поліноміальна регресія, алгоритм градієнтного спуску за міні-батчами, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 101)

   num_coef=4
   coef=[-100,2,1,100]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*20.5
   ```

   (б) `sklearn.datasets.fetch_california_housing`

2. Лінійна регресія, алгоритм градієнтного спуску, MSE.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=3
   coef=[1,2,3]
   ```

```
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*0.5
```

(б) `sklearn.datasets.load_diabetes`

3. Поліноміальна регресія, алгоритм градієнтного спуску з моментом, MSE.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=5
   coef=[10,2,30,4,5]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*1.5
   ```

   (б) `sklearn.datasets.load_boston`

4. Поліноміальна регресія, алгоритм Adam, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=9
   coef=[1,2,3,3,5,6,4,300,2]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*20.5
   ```

   (б) `sklearn.datasets.make_friedman3`

5. Лінійна регресія, алгоритм градієнтного спуску, MSE.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 101)

   num_coef=3
   coef=[-10,2,3]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*1.5
   ```

(б) `sklearn.datasets.load_boston`

6. Поліноміальна регресія, алгоритм градієнтного спуску, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=8
   coef=[1,20,3,4,6,7,300,2]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*20.5
   ```

   (б) `sklearn.datasets.make_friedman3`

7. Поліноміальна регресія, алгоритм градієнтного спуску з моментом, MSE з регуляризацією за нормою L_1. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=5
   coef=[10,2,30,4,5]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*1.5
   ```

   (б) `sklearn.datasets.load_boston`

8. Поліноміальна регресія, алгоритм Adam, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 100)

   num_coef=7
   coef=[1,2,3,30,5,6,4]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*3.5
   ```

(б) `sklearn.datasets.fetch_california_housing`

9. Поліноміальна регресія, алгоритм градієнтного спуску з моментом, MSE з регуляризацією за нормою L_1. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

   Початкові дані:

   (а)
   ```
   X_data = np.linspace(-1, 1, 101)

   num_coef=10
   coef=[-100,2,3,-3000,5,6,4,300,2,600]
   y_data=0
   for i in range(num_coef):
     y_data += coef[i]*np.power(X_data, i)
   y_data += np.random.randn(*X_data.shape)*100.5
   ```

   (б) `sklearn.datasets.make_friedman2`

10. Лінійна регресія, алгоритм градієнтного спуску, MSE.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 101)

    num_coef=3
    coef=[-10,2,3]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*1.5
    ```

    (б) `sklearn.datasets.load_boston`

11. Лінійна регресія, алгоритм градієнтного спуску, MSE.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 100)

    num_coef=2
    coef=[10,3]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*1.5
    ```

    (б) `sklearn.datasets.load_diabetes`

12. Поліноміальна регресія, алгоритм градієнтного спуску за міні-батчами, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

    Початкові дані:

    (а)
    ```
    def make_data(N, err=1.0, rseed=1):
        rng = np.random.RandomState(rseed)
        X = rng.rand(N, 1) ** 2
        y = 10 - 1. / (X.ravel() + 0.1)
        if err > 0:
            y += err * rng.randn(N)
        return X, y

X, y = make_data(200)
    ```

    (б) `sklearn.datasets.fetch_california_housing`

13. Поліноміальна регресія, алгоритм Adam, MSE з регуляризацією за нормою L_1. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

    Початкові дані:

    (а)
    ```
X_data = np.linspace(-1, 1, 101)

num_coef=10
coef=[-100,2,3,-3000,5,6,4,300,2,600]
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*100.5
    ```

    (б) `sklearn.datasets.make_friedman2`

14. Лінійна регресія, алгоритм градієнтного спуску, MSE.

    Початкові дані:

    (а)
    ```
X_data = np.linspace(-1, 1, 100)

num_coef=3
coef=[1,2,3]
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*0.5
    ```

    (б) `sklearn.datasets.load_diabetes`

15. Поліноміальна регресія, алгоритм градієнтного спуску за міні-батчами, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 100)

    num_coef=5
    coef=[10,2,30,4,5]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*1.5
    ```

    (б) `sklearn.datasets.fetch_california_housing`

16. Лінійна регресія, алгоритм градієнтного спуску, MSE.

    Початкові дані:

    (а)
    ```
    def make_data(N, err=1.0, rseed=1):
            rng = np.random.RandomState(rseed)
            X = rng.rand(N, 1) ** 2
            y = 10 - 1. / (X.ravel() + 0.1)
            if err > 0:
                y += err * rng.randn(N)
            return X, y

    X, y = make_data(200)
    ```

    (б) `sklearn.datasets.load_diabetes`

17. Лінійна регресія, алгоритм градієнтного спуску, MSE.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 101)

    num_coef=2
    coef=[-1,3]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*1.5
    ```

    (б) `sklearn.datasets.load_boston`

18. Поліноміальна регресія, алгоритм Adadelta, MSE з регуляризацією за нормою L_2. Дослідити різні значення параметра регуляризації і підібрати найкраще з них.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 100)

    num_coef=7
    coef=[1,2,3,30,5,6,4]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*3.5
    ```

    (б) `sklearn.datasets.fetch_california_housing`

19. Поліноміальна регресія, алгоритм Adadelta, MSE.

    Початкові дані:

    (а)
    ```
    def make_data(N, err=1.0, rseed=1):
            rng = np.random.RandomState(rseed)
            X = rng.rand(N, 1) ** 2
            y = 10 - 1. / (X.ravel() + 0.1)
            if err > 0:
                y += err * rng.randn(N)
            return X, y

    X, y = make_data(200)
    ```

    (б) `sklearn.datasets.load_diabetes`

20. Лінійна регресія, алгоритм Adam, MSE.

    Початкові дані:

    (а)
    ```
    X_data = np.linspace(-1, 1, 100)

    num_coef=5
    coef=[10,20,3,40,5]
    y_data=0
    for i in range(num_coef):
      y_data += coef[i]*np.power(X_data, i)
    y_data += np.random.randn(*X_data.shape)*2.5
    ```

    (б) `sklearn.datasets.load_boston`

21. Лінійна регресія, алгоритм Adagrad, MSE.

    Початкові дані:

(а)
```
X_data = np.linspace(-1, 1, 101)

num_coef=4
coef=[-10,2,30,5]
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*3.5
```

(б) `sklearn.datasets.load_diabetes`

22. Поліноміальна регресія, алгоритм градієнтного спуску з моментом, MSE.

Початкові дані:

(а)
```
X_data = np.linspace(-1, 1, 100)

num_coef=9
coef=[1,2,3,3,5,6,4,200,2]
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*20.5
```

(б) `sklearn.datasets.make_friedman3`

23. Поліноміальна регресія, алгоритм Adam, MSE.

Початкові дані:

(а)
```
X_data = np.linspace(-1, 1, 101)

num_coef=10
coef=[-100,2,3,-3000,5,6,4,300,2,600]
y_data=0
for i in range(num_coef):
  y_data += coef[i]*np.power(X_data, i)
y_data += np.random.randn(*X_data.shape)*100.5
```

(б) `sklearn.datasets.make_friedman2`

3 Контрольні питання для захисту роботи

- Навести особливості бібліотеки TensorFlow.
- Характеристики бібліотек глибокого навчання з відкритим кодом: Deeplearning4j, Caffe, Theano, Torch.

- Що таке граф обчислень. Навести приклад. Як створити граф обчислень в TensorFlow?
- Які основні переваги створення графа обчислень замість виконання обчислень безпосередньо? Які головні недоліки?
- Як виконати граф обчислень? Навести приклад.
- Коли змінна ініціалізується? Коли вона знищується?
- Які є способи ініціалізації змінних в TensorFlow?
- Векторизація обчислень.
- Життєвий цикл значення вузла графу обчислень.
- Як реалізовано метод ґрадієнтного спуску в TensorFlow ?
- Як передаються дані алгоритму навчання?
- Як зберегти і відновити модель навчання в TensorFlow?
- Спільне використання змінних. Які способи розділу змінних між різними компонентами графа обчислень?