Spring 2024:CS5720

Neural Networks and Deep Learning - ICP-1

Gonaboyina Vijay Vardhan (700755141)

Github link: https://github.com/Vijayvardhan02/Neural-network-deep-learning-ICP2

Video:https://drive.google.com/file/d/11oyXwjFtxbCDKRZ2ifh6g84OeHp6U2B4/view?usp=drive_link

1.Create a class Employee and then do the following

- Create a data member to count the number of Employees
- Create a constructor to initialize name, family, salary, department
- Create a function to average salary • Create a Fulltime Employee class and it should inherit the properties of Employee class
- Create the instances of Fulltime Employee class and Employee class and call their member functions.

Code :

```
class Employee:
    # Class variable to count the number of employees
    employee_count = 0

    def __init__(self, name, family, salary, department):  # Changed _init_ to __init__
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.employee_count += 1

    def average_salary(self, employees):
        total_salary = sum(employee.salary for employee in employees)
        return total_salary / len(employees) if employees else 0

    def __str__(self):  # Changed _str_ to __str__
        return f"Employee(name={self.name}, family={self.family}, salary={self.salary}, department={self.department})"


class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department):  # Changed _init_ to __init__
        super().__init__(name, family, salary, department)
        # attributes to FulltimeEmployee can be added here

    def __str__(self):  # Changed _str_ to __str__
        return f"FulltimeEmployee(name={self.name}, family={self.family}, salary={self.salary}, department={self.department})"
```

```python
# Creating instances of Employee
emp1 = Employee("vijay", "Gonaboyina", 60000, "HR")
emp2 = Employee("Ravi", "Indugula", 70000, "Finance")

# Creating instances of FulltimeEmployee
ft_emp1 = FulltimeEmployee("Prem", "Manyam", 80000, "IT")
ft_emp2 = FulltimeEmployee("Swaroop", "Adabala", 75000, "Marketing")

# List of all employees
all_employees = [emp1, emp2, ft_emp1, ft_emp2]

# Calculate and print average salary(average of salary of all employees)
employee = Employee("Temp", "Temp", 0, "None")
print(f"Average Salary: {employee.average_salary(all_employees)}")

# Print the details of each employee
for emp in all_employees:
    print(emp)

# Print total number of employees
print(f"Total number of employees: {Employee.employee_count}")
```

OUTPUT:

```
Average Salary: 71250.0
Employee(name=vijay, family=Gonaboyina, salary=60000, department=HR)
Employee(name=Ravi, family=Indugula, salary=70000, department=Finance)
FulltimeEmployee(name=Prem, family=Manyam, salary=80000, department=IT)
FulltimeEmployee(name=Swaroop, family=Adabala, salary=75000, department=Marketing)
Total number of employees: 5
```

2.Numpy : Using NumPy create random vector of size 20 having only float in the range 1-20. Then reshape the array to 4 by 5 Then replace the max in each row by 0 (axis=1) (you can NOT implement it via for loop)

Code:

```python
[ ]
    import numpy as np

    #random vector of size 20 with floats in the range 1-20
    vector = np.random.uniform(1, 20, size=20)
    print(vector)
    print()

    #vector to 4x5
    matrix = vector.reshape(4, 5)

    #Replace the max in each row with 0 (axis=1)
    # Get the indices of the maximum values along axis=1
    max_indices = np.argmax(matrix, axis=1)

    # Create a boolean mask where the max values are True
    mask = np.arange(matrix.shape[1]) == max_indices[:, None]

    # Apply the mask to set max values to 0
    matrix[mask] = 0

    # Output the result
    print("Matrix after replacing max values with 0:")
    print(matrix)
```

OUTPUT:

```
[ 5.7288541  19.03254357  5.04850547 11.72626289 12.39914445  3.68495317
 10.17231779  1.18531313 10.31414552  2.68202525 19.46944046 19.16499015
  4.31400588 10.70520048 14.96850066 15.55112818 10.7125966   4.20562257
 12.52663931  6.07081294]

Matrix after replacing max values with 0:
[[ 5.7288541   0.          5.04850547 11.72626289 12.39914445]
 [ 3.68495317 10.17231779  1.18531313  0.          2.68202525]
 [ 0.         19.16499015  4.31400588 10.70520048 14.96850066]
 [ 0.         10.7125966   4.20562257 12.52663931  6.07081294]]
```