Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-3

NAME:NANDIMANDALAM VENKATA VINAY VARMA

STUDENT ID:700745193

Github Link: https://github.com/venkatavinayvarma/NeuralNetworks_ICP3.git

Video Link: https://drive.google.com/drive/folders/1B0X1eq38WGeVXGh2-kyPpdM1e71SFWM5?usp=sharing

In class programming:

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.

```
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.
```

[4]:
```
class Employee:
    noOfEmployees = 0
    salarySum =0
    # Initializes an Employee object.

    def __init__(self, name, family, salary, department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.noOfEmployees += 1
        Employee.salarySum += self.salary

    # Calculates the average salary of all employees.
    def average_salary(self):
        return sum([Employee.salarySum for emp in employees])/Employee.noOfEmployees

    # This class represents a full-time employee, who inherits from the Employee class and has an additional bonus attribute
class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department, bonus):
        super().__init__(name, family, salary, department)
        self.bonus = bonus

    # Create employee instances
employee1 = Employee("vinay", "Varma", 10000, "IT")
employee2 = FulltimeEmployee("vikas", "Varma", 80000, "HR", 1000)
employee3 = FulltimeEmployee("Raju", "Varma", 70000, "Marketing",500)
employees = [employee1, employee2, employee3]

    # Print the number of employees
print(f"Number of Employees: {Employee.noOfEmployees}")

    # Print the average salary
print(f"Average Salary: ${employee1.average_salary():.2f}")
```

```
    # Print the number of employees
print(f"Number of Employees: {Employee.noOfEmployees}")

    # Print the average salary
print(f"Average Salary: ${employee1.average_salary():.2f}")

    # Print individual employee salaries
print(f"{employee1.name}'s salary: ${employee1.salary:.2f}")
print(f"{employee2.name}'s salary: ${employee2.salary + employee2.bonus:.2f}")
print(f"{employee3.name}'s salary: ${employee3.salary + employee3.bonus:.2f}")
```

```
Number of Employees: 3
Average Salary: $160000.00
vinay's salary: $10000.00
vikas's salary: $81000.00
Raju's salary: $70500.00
```

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)

```
2)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)
```

```
[6]: import numpy as np # Importing numpy library
vector_size = np.arange(1,21,dtype=float) # Creating a numpy vector with arange function
print(vector_size)
vector_size=vector_size.reshape(4,5)     # Reshaping the vector with reshape() function
print(vector_size)
vector_size=np.where(np.isin(vector_size,vector_size.max(axis=1)),0,vector_size)# Finding the max values in row and replacing them with zero using np.whe
vector_size
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
 19. 20.]
[[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]
 [11. 12. 13. 14. 15.]
 [16. 17. 18. 19. 20.]]
```

```
[6]: array([[ 1.,  2.,  3.,  4.,  0.],
       [ 6.,  7.,  8.,  9.,  0.],
       [11., 12., 13., 14.,  0.],
       [16., 17., 18., 19.,  0.]])
```