Forelesning7

October 24, 2023

1 Lecture 6: Inheritance

1.1 Classes: ways to define your own data types

[ ]:

```
# example of a simple class object. this stores the information about an apple

class apple:
    def __init__(self, color, flavor):
        self.color = color
        self.flavor = flavor
```

1.2 Terminology

1. Data type: how to interpret chunks of memory
2. Object: A chunk of data given an address in computer memory. An object can be created, copied, deleted, and passed around in a program.
3. Constructor: a method that is called when an object is created
4. Class: A class is a user-defined data type. It specifies

- Which data an object contains
- operations that may be perfomed on the data

5. Instance: an object of a class
6. Method: a function that is associated with a class
7. data attribute: a variable that is associated with a class
8. Overloading: defining a method with the same name as an existing method, but with different arguments
9. Abstract classes: classes that cannot be instantiated

[8]:

```
import numpy as np
class NeuralNetwork:
    def __init__(self,layers):
        self.size = len(layers)
        self.layers = layers
        self.activation = lambda x: self.tanh(x)
        self.generate_weights_and_biases()

    def relu(self,x):
        return np.maximum(0,x)

    def sigmoid(self,x):
        return 1/(1+np.exp(-x))

    def tanh(self,x):
        return np.tanh(x)

    def generate_weights_and_biases(self):
        self.weights = [np.random.rand(rows,cols) for cols,rows in zip(self.
 ↪layers[:-1],self.layers[1:])]
        self.biases = [np.random.rand(rows) for rows in self.layers[1:]]

    def forward(self,x):
        for w,b in zip(self.weights,self.biases):
            x = self.activation(w @ x + b)
        return x

layers = [64,128,128,128,10]

input_vector = np.random.rand(layers[0])

print(NeuralNetwork(layers).forward(input_vector))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

1.3 Abstraction

[ ]:

```
from abc import ABC, abstractmethod

class Animal(ABC):
    def __init__(self, name):
        self.name = name

    @abstractmethod
    def make_sound(self):
        pass

# when creating a child class of this "animal" we need to implement the␣
 ↪abstract method "make_sound"
```

1.4 Why object oriented?

1.4.1 Main benefit

- Modularity: the ability to break a program into self-contained pieces

- Encapsulation: the ability to hide the details of an object from the rest of the program
- Polymorphism: the ability to use the same syntax for different types of objects

1.4.2 Disadvantages

- Overhead: object oriented programs are often slower than procedural programs
- Complexity: object oriented programs are often more complex than procedural programs
- Rigidity: object oriented programs are often more difficult to modify than procedural programs