Cognitive Architectures Assignment 4

Connectionism

Task 1.1:

Connectionism is a movement in cognitive science that hopes to explain intellectual abilities using artificial neural networks. Connectionist models provide a new paradigm for understanding how information might be represented in the brain. The artificial neural networks consists of a large number of units connected in a "net"(a pattern of connections). These connections are weighted, and its by adjusting the weights that the neural network learns.

A weakness in connectionist approaches is that most neural network research abstracts away from many interesting and possibly important features of the brain. It is widely felt that neural networks are not particularly good at the kind of rule-based processing needed for language, reasoning and higher forms of thought.

Artificial Neural Networks

Task 1.3:

The statement that artificial neural networks are universal function approximators means that a artificial neural network can approximate almost any known function. The Universal Function Approximation theorem states that this can be done with a feed-forward net containing a single hidden layer with a finite number of neurons.

Task 1.4:

Image classification can be addressed by a universal function approximator because images consists of pixels. The color of each pixel can be represented by numerical values, and such the combination of these values for all pixels in an image can be used in a mathematical model for image classification.

Task 1.6:

The sigmoid activation function and the ReLU activation function are two commonly used activation functions for neural nets. The sigmoid function is commonly used when we must predict the probability of an output, that is because it maps any input to a value in the range 0 to 1. The sigmoid function will map very negative values to approximately 0, and very positive values to approximately 1.

The ReLU is an activation function that maps any negative value to 0, and mapping positive values to their own value, not a maximum of 1. This makes the ReLU function excellent for learning quickly. The ReLU function is the most commonly used activation function for deep neural networks.

Task 1.7:

The “association black box” analogy of artificial neural networks is an analogy based on neural networks being black boxes. A “black box” is a device which can be viewed only by its inputs and outputs, without knowing any of the inner workings of this box. In short this means that since we do not quite understand all of what is going on “under the hood” we can never truly understand how artificial neural networks function.

Perceptron

Task 1.10:

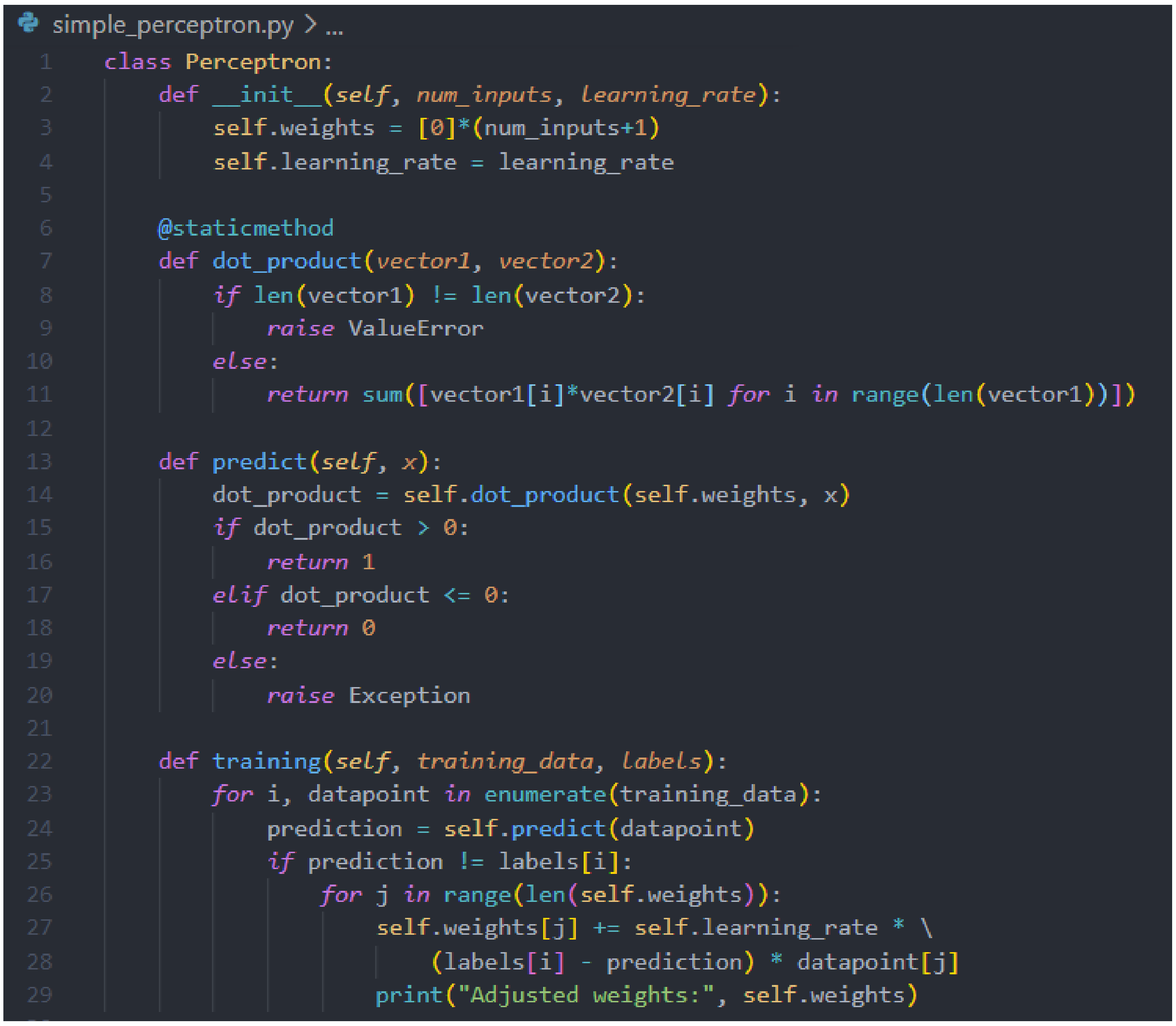
Binary classification is an example of a problem that can be solved by a perceptron. Binary classification is the task of classifying elements of a given set into two categories based on given rules. Rosenblatt perceptron can only handle binary classification for linearly separable classes.

Task 1.11:

A nonlinearly separable classification problem cannot be solved by a perceptron, that is unless it has preprocessed the inputs in order to recast it into a linearly separable problem

Task 1.12:

Here is the code for my python implementation of the perceptron:



simple_perceptron.py > ...

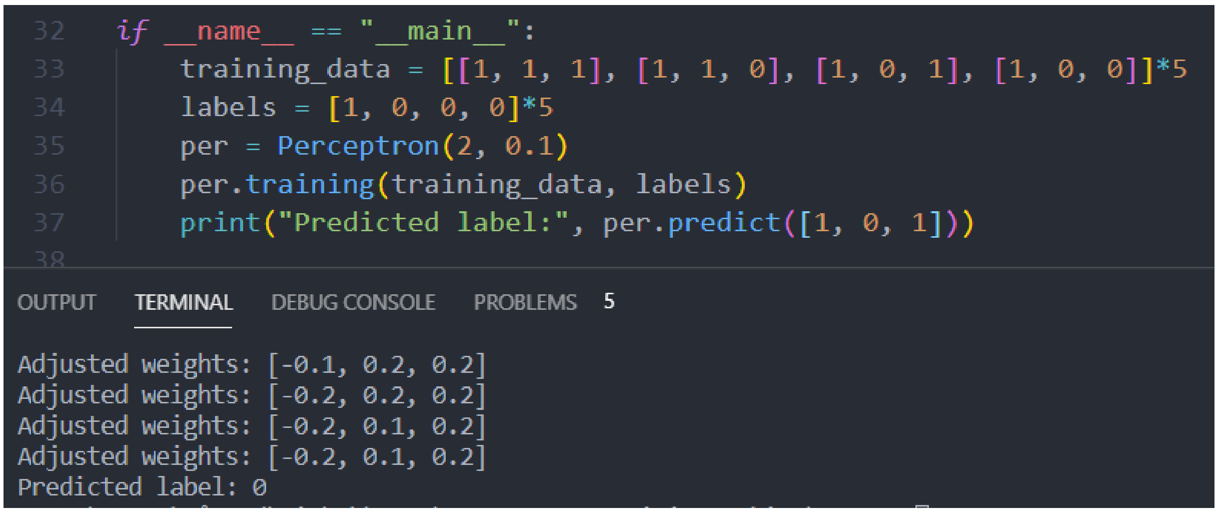
```python
class Perceptron:
    def __init__(self, num_inputs, learning_rate):
        self.weights = [0]*(num_inputs+1)
        self.learning_rate = learning_rate

    @staticmethod
    def dot_product(vector1, vector2):
        if len(vector1) != len(vector2):
            raise ValueError
        else:
            return sum([vector1[i]*vector2[i] for i in range(len(vector1))])

    def predict(self, x):
        dot_product = self.dot_product(self.weights, x)
        if dot_product > 0:
            return 1
        elif dot_product <= 0:
            return 0
        else:
            raise Exception

    def training(self, training_data, labels):
        for i, datapoint in enumerate(training_data):
            prediction = self.predict(datapoint)
            if prediction != labels[i]:
                for j in range(len(self.weights)):
                    self.weights[j] += self.learning_rate * \
                        (labels[i] - prediction) * datapoint[j]
                print("Adjusted weights:", self.weights)
```

Here is an example of a test run using the AND function:



```python
if __name__ == "__main__":
    training_data = [[1, 1, 1], [1, 1, 0], [1, 0, 1], [1, 0, 0]]*5
    labels = [1, 0, 0, 0]*5
    per = Perceptron(2, 0.1)
    per.training(training_data, labels)
    print("Predicted label:", per.predict([1, 0, 1]))
```

```
Adjusted weights: [-0.1, 0.2, 0.2]
Adjusted weights: [-0.2, 0.2, 0.2]
Adjusted weights: [-0.2, 0.1, 0.2]
Adjusted weights: [-0.2, 0.1, 0.2]
Predicted label: 0
```

We see that the predicted label is correct. The implementation works for all cases of both AND and OR function. For training data I provide all the possible combinations of the function, 5 times. I do this 5 times because my implementation was struggling with adjusting with only 4 points of training data but adjusting the learning rate might also give the same result.

General discussion

Task 1.16:

Artificial Neural Networks have no notion of symbols and hierarchical representations of knowledge, but are based solely on mathematics and probability functions. Deep learning neural networks use a set of techniques that allows the system to discover the representations needed for feature detection or classification from raw data.

Task 1.18:

In artificial neural networks there are things that function similarly to how production rules function, but the main difference is how these "rules" are achieved. Artificial neural networks can learn from its input data and approach a state in which it cant be told apart from a rule-based AI.

Task 1.19:

Generally speaking, connectionists claim that knowledge in the human brain is stored non-symbolic in the weights or connection strengths between the brain neurons. There are however some connectionists that argue that the brain's neural net indeed implements a symbolic processor.

Belief-Desire-Intention architecture

Task 1.21:

The Belief-Desire-Intention architecture are based on different architecture components, such as:

- Beliefs, which represents the informational state of the agent. Beliefs are stored in a database as a Beliefset.
- Desires, which represents the motivational state of the agent. A goal is a desire which has been selected by the agent for pursuing.

- Intentions, which represents the deliberative state of the agent, or what the agent has chosen to do. A plan is a sequence of actions that an agent can perform in order to achieve one of its intentions.
- Events, which are triggers for reactionary actions by the agent. An event may trigger actions, modify goals, or update beliefs.

<u>Task 1.22:</u>

The theory of “practical reasoning” is a theory of human reasoning developed by Michael Bratman. This theory is a way of explaining future-directed intention, based on the notion that human mental models of the world are indeed theories themselves. The “practical reasoning” theory was used to develop the Belief-Desire-Intention Architecture.

Subsumption

<u>Task 1.23:</u>

The subsumption architecture is a behavior-based reactive robotic architecture. Instead of guiding behavior by symbolic mental representations of the world, the subsumption architecture couples sensory data to action selection in a bottom-up approach. The architecture does this by decomposing the complete behavior into sup-behaviors, organizing these sub-behaviors into a hierarchy of layers. These layers all receive sensory information, and work in parallel to generate outputs to be used by the actuators.

<u>Task 1.25:</u>

The layered structure of the subsumption architecture is a hierarchy of layers in which higher layers can subsume lower layers to create viable behavior. These layers all receive sensory information, and work in parallel to generate outputs to be used by the actuators.

Hybrid Architectures

<u>Task 1.26:</u>

Hybrid architecture have the benefit of combining both reactive and deliberative sub-components. In hybrid architectures the reactive component is often given a precedence over the deliberative one. These architectures aim to balance proactiveness and reactivity. The reactive component would be able to respond to world changes without any complex reasoning and decision-making. The deliberative sub-system would be responsible for abstract planning and decision-making using symbolic representations.