Kogark superduper assignment ark

Assignment 2

> **Task 1.1 3p**: Why do we separate methods in symbolic methods and sub-symbolic methods? Explain the difference. Why is it customary to use the term 'sub-symbolic' and not 'non-symbolic'?

Sander:

We separate the methods because they operate with different types of data. Symbolic approaches operate with strings of characters that represent real-world entities or concepts. They take the symbols, applies rules and spit out an output. Sub-symbolic methods can work with numerical patterns and/or probabilistic data, a more distributed representation, such as a neural network or CDs.

We use the term "sub-symbolic" and not "non-symbolic" because even though the system is sub-symbolic, it does operate with symbols. Non-symbolic suggests that there are no symbols used at all.

Marius:

Having information about the world represented in symbolic methods has two core benefits. One is that you can easily combine symbols stored by this knowledge. This is because they are easily comparable, and you know what kind of characteristics the symbols have. The second benefit is that when symbols are easily comparable, it makes it possible for a cognitive agent to reflect upon their behavior.

If we look at the “chinese room argument” there are two outcomes. Let’s say we want to translate a text from English to mandarin. If the man in the room represents a symbolic AI, the man will look at the different information located in the room and return a reasonable output. If the man in the room represents a non-symbolic AI, the man has already learned how to translate from English to Chinese.

In the second case, the AI has been trained and uses sub-symbolic values to translate the text.

It is customary to use sub-symbolic, since the computer uses symbols that are learned, like a neural network. Non-symbolic refers to not having any symbols at all, which is kind of misleading.

> **Task 1.2 2p**: What is the main difference between cognitivistic and emergent approaches?

Sander

The main difference between cognitivistic and emergent approaches is that a cognitivistic process is a symbolic system, whereas emergent systems are sub-symbolic. A cognitivistic approach is more like "if-then" rules. Like the previous question - it applies rules to symbols and spits out an output. Emergent systems are more dynamic. It can learn and adapt, by building massively parallel models, such as neural networks, where information is, like in neurons, propagation of signals from input nodes.

Marius

An emergent approach is an approach that develops over time. (sub-symbolic)When a system is exposed to different factors. This behavior is based on a set of data that is possible to combine in order to behave in a more complex manner. It achieves predictable global effects.

A cognivistic approach(symbolic system) is when an AI relies on algorithms to solve a problem or be able to find patterns. This approach often consists of trying to mimic the reasoning process to solve a set of problems as the problems change.

The ultimate goal of an emergent cognitive system is to maintain its own autonomy, and cognition is the process by which it accomplishes this.

> **Task 1.3 2p**: If a system shows intelligent behaviour through emergent approaches, what does this mean?

This means that the emergent system is thinking rationally and are *learning* from the problems it is solving.

1.1.2 Hybrid architectures

> **Task 1.4 2p**: What are some key benefits of creating hybrid architectures? Why do people create this type of architecture?

The benefits of creating hybrid systems are that we get "the best of both worlds". We do not, for instance, have to explicitly program all knowledge, like in a cognivistic system, because emergent systems are more dynamic and can adapt and learn. This makes the system able to understand the external world.

> **Task 1.5 2p**: When combining symbolic and sub-symbolic systems to create a hybrid architecture, what is included from each?

From sub-symbolic systems, at least one module for sensory processing are included, and the rest of the knowledge and processing are included from symbolic systems.

1.2 Perception and sensing

> **Task 1.6 2p**: Why is cognitive architecture research very often centred around vision?

Regardless of what an intelligent system is meant to do, it will need perception - that is, input from the outside world to produce behaviour. *Vision* is the dominating way of perceiving the 'outside world' and are therefore extremely central in cognitive architecture.

> **Task 1.7 2p**: Explain the three stages of vision as proposed by David Marr.

Marr proposed visual processing as three stages, **Early Vision**, **Intermediate vision** and **Late stage vision**.

In the Early vision stage, the system is perceiving simple elements such as colour, luminance, shape and motion. In the intermediate stage, these simple elements are grouped into regions, which in the late stage are recognized as objects and assigned meaning based on the knowledge the system already has on the environment.

> **Task 1.8 2p**: What applications is audition used for in cognitive architectures?

Audition is mostly used when the system has to precept sound. This is widely used when a system wants to be a text-to-speech (or speech-to-text) system, such as, for example, google translate or a Google home/Alexa.

1.3 Attention

> **Task 1.9 3p**: Explain the three classes of information reduction mechanisms and how they relate to each other

Attention is closely related to perception. It is what you do after you have perceived.

The three classes of information reduction mechanisms are **selection**, **restriction** and **suppression**.

Selection focuses on choosing one from many. That is - deciding what object/event to focus on or what region/feature/time/object, etc.. are of interest.

Restriction restricts the amount of information you have. I.e. choose some from many to focus on. Another way to put it is that it prunes the search space. This is done by preparing the visual system for input based on task demands, the domain knowledge, external stimuli, restrict attention to objects relevant for the task and limit the field of view. Restriction uses the goal of the task to restrict the search space to only relevant information, based on the system's knowledge.

Suppression suppresses some from many. This means that there is much noise that is not relevant to the current goal, such as a TV playing in the background or other irrelevant sound.

> **Task 1.10 2p**: Explain the difference between data-driven and task-driven attention. Use at least one example.

Data-driven tries to identify the most noticeable or interesting regions in an image. For example, in a picture of a snowy field and a req glove, the region containing the glove would be identified, rather than the rest of the image, because of it's colour and shape. A task-driven approach would also focus this glove but in a different manner. It would have been told that it is looking for a glove or a red object.

1.4 Action selection

> **Task 1.11 3p**: What are the two major approaches to action selection, and how does this relate to symbolic vs. non-symbolic architectures?

Action selection determines "what to do next?". This is done in two major approaches, **Planning** and **Dynamic**. In the planning approach, the system plan all the steps it will do in front. In the dynamic approach, each step is calculated "in the moment" - the best action is chosen among the alternatives based on the available knowledge.

This can be related to symbolic and non-symbolic architectures in the sense that symbolic arcitechtures are parsing symbols, and planning action selection is a predefined set of actions that are waiting to be parsed, wheras dynamic action selection are "on the go", such as non-symbolic systems are "on the go" - like a self moving car.

1.5 Memory

> **Task 1.12 2p**: Explain the multi-store concept of memory.

The multi-store concept is a structural model proposed by Atkinson and Shiffrin in 1968. It says that memory consists of three stores: a sensory register, short-term memory and long-term memory.

> **Task 1.13 2p**: It is common to distinguish between short-term and long-term memory. From which research discipline does this separation come from?

This discipline comes from the study of psychology.

> **Task 1.15 2p**: Mention some key differences in how knowledge is represented in symbolic vs. non-symbolic architectures.

Knowledge is represented in lots of different ways. In symbolic architectures, the most common representation of knowledge is "if-then" statements. These are often represented as graphs (or flow charts). In emergent systems and non-symbolic architectures, however, the most common representation is by a set of neural networks or state/action pairs, who holds input/output values.

2 Soar

> **Task 1.16**: Explain the following phases: Input phase, Operator selection, and Operator application.

Input phase, here the memory represents the perception of the system. It uses, for example, its vision to determine what kind of states that the system should have.

Operator selection, during the operator selection, the system determines what kind of operation to perform. This phase consists of two actions: Elaboration and decision. In this phase, the system compares the states gathered from the input phase with corresponding rules in its memory. In that way, it can decide what kind of operation should be performed. In the decision part of the Operator selection, it uses the long term and working memory to generate an action.

Operator application, After selecting an operator, the system has to apply the new operation. Elaboration. The application part of the system produces commands. And the output is the results of the commands from the previous step.

Assignment 3

> **Task 1.1** 3p: Explain the notion of problem space in Soar.

The tasks in SOAR are represented as a collection of problem spaces. These problem spaces are made up of a state and operators that can be used to manipulate the states.

When SOAR wants to begin a task, it chooses one of these problem spaces and an initial statie in the space. Then, a single operator from the problem space are selected and applied to the agent's current state, leading to internal changes such as retrieval of knowledge, modifications or external actions. This is repeated until the task's *goal state* are reached.

> **Task 1.2 3p**: What is an '*impasse*', and how does Soar handle the situation?

An *impasse* is a situation where when the knowledge to select or apply an operator is incomplete or uncertain. When an impasse happens, SOAR aitomatically creates a **sub-state** to the current state. A **sub-state** is a state where a temporary goal are created. This goal are concerned with retrieving or discovering knowledge, so that the decision making can continue. This process of problem solving are used recursively, which can lead to a stack of sub-states.

> **Task 1.3: 3p** optional Explain two or more different impasse types.

An impasse is a possibility for learning, and for the system to provide better action, it will use several impasse processes.

- *Chunking* is an impasse type. This consists of evaluating different operators so that the system can provide a context that triggers a long term memory(LTM) action.
- *Reinforcement learning* is when the system gets a reward from the environment so that it knows whether or not the action is correct. This consists of two steps; having knowledge about action selections, and be able to select operators and how to understand the reward system.
- *Episodic memory* is a source of knowledge that is available in the stream of the systems experience. This problem-solving is by using experience on actions that have worked before, in order to provide an action for a solution now. An example of this is if a car mechanic has changed the oil filter before but never on a given car model. Here the mechanic uses earlier experiences to determine how to change the oil filter.

> **Task 1.4 2p**: Describe the different ways Soar can learn from its problem solving experience.

It can learn in the *Reinforcement learning cycle*. The goal of this cycle is to "*learn an action-selection policy such as to maximize expected reciept of future reward.*". Reinforcement learning changes the rules in the procedural memory, that is, a type of long term memory, where knowledge are encoded as rules. So, in the reinforcement cycle, an action will be taken, and rewarded based on how good the action was.

Another way to learn are through *Semantic learning*, where it learns from capturing declarative statements (or facts) and through *Episodic learning*, where the architecture stores history of experiences.

> **Task 1.5 3p**: Explain the 'symbol structures' that exist in Soar. Explain processes that change these symbol structures over time

SOAR operates with states and operators. The state stores information about the system's experiences, e.g through perception, and the operators consists of information on what action(s) to take when a new problem is encountered. The processes that changes these structures are for example thorugh perception to change the state, and through learning (previuosly explained) to change the operators.

1.2 Memory

> **Task 1.6 2p**: Describe the three types of long term memory (LTM) structures in Soar and their roles in problem solving

There are three types og LTM in SOAR: Procedual, semantic and episodic. Theese three types of memory have different roles in problem solving. The procedual memory are accessed directly during the decision cycle, and the semantic and episodic memory are accessed deliberately through the creation of specific cues in the working memory,

> **Task 1.8 3p**: How does the WM and LTM communicate / cooperate?

The working memory and long term memory work together in the sense that the contents of the working memory trigger retrieval from the LTM. That is, when the working memory percepts something, it triggers a retrieval from the long term memory to get information of the problem it's currently having. This is done through the three-phase decicion cycle. First, the WM inputs the elements it has, then the situation is elaborated though matching of the "if" parts of the rules in LTM and operators are evaluated based on their ability to be usefull in the current situation. Lastly, in the decision step, an operator are selected.

1.4 Practical

> **Task 1.10 3p**: Write the following rule (presented in plain language) as a Soar production rule.

```
if we are in the ball-in-net problem space AND
we desire to get the ball in the net AND
the current state says the ball is not in the net
then propose an operator to apply to the current state AND
call this operator ’kick ball ’
```

The rule as SOAR production rule:

```
sp {ball-in-net*propose*kick-ball
    (state <s> ^problem-space <p> ^desired <d>)
    (<p> ^name ball-in-net)
    (<d> ^ball-in-net yes)
    (<s> ^ball-in-net no)
    -->
    (<s> ^operator <o>)
    (<o> ^name kick-ball)}
```

2 ICARUS

> **Task 1.11 3p**: Describe the conceptual inference process in ICARUS.

In the ICARUS architecture, conceptual inference is a way to interperet the environment around the system. On each cognitive cycle, ICARUS' conceptual inference module collects all percepts comming from the environment, finds consistent matches of conceptual clauses (symbolic predicates that the agent can use for this cognitive purpose) against these percepts, and infers beliefs (instances of concepts that describe the agent's situation) that correspond to these rules' heads and adds them to the belief memory.

> **Task 1.13 3p**: How can ICARUS learn from its problem solving behavior?

ICARUS learns by previuosly solved problems. It usually achieves a goal by chaining off a skill clause. Whenever a new goal are reached, the skill are added to the skill clause as a subskill. The next time the system encounteres the same problem, it has the skill to reach the goal in the skill clause.

> **Task 1.15 3p**: In ICARUS, the LISP programming language is used to create concepts, rules, and beliefs. LISP uses something that is known as prefix notation, explain what this is.

Prefix notation in LIPSP are also known as Cambridge Polish Notation. It means that the operator are put in front of the operands, rather than between them. For example:

`1+2*3+3+4` is the same as `( + (+ 1 ( * 2 3) ) 4 )` which can be interpreted as `((2*3) + 1) + 4`. (You start with the inner parentheses).

> **Task 1.16 3p**: Provide at least one example of a syntactically valid clause in LISP which expresses a reasonable statement related to the real world which is not appearing in the lecture slides or the

> recommended papers.

Example of a valid LISP clause for calculating fibonacci numbers:

```
(defun fibonacci (N)
  (if (or (zerop N) (= N 1))
      1
    (+ (fibonacci (- N 1)) (fibonacci (- N 2)))))
```

3 ACT-R

> **Task 1.17 3p**: How does ACT-R relate to the concept of symbolic vs. sub-symbolic cognitive architectures?

ACT-Rs production system is symbolic and it has a subsymbolic structure that is a set of processes that can be summarized by a number of mathematical equations. These subsymbolic equations control many of the symbolic processes, such as estimating a relative cost and benefit associated with different productions rules.

> **Task 1.19 3p**: How does learning work in ACT-R?

Learning in ACT-R works in two different ways. Instance learning and utility learning.

Instance learning is learning by retrieving old experiences from the memory and *utility learning* is learning which of multiple available stratigies is optimal, by keeping track of costs and probabilty of success.

In instance learning, the architecture makes use of so called *chunks*. These chunks contain facts, such as "Oslo is the capital of Norway" or "1+1=2". When encountering a problem, a chunk that fits the current situation, and all related chunks are activated. Later, the odds of this chunk is chosen is higher than before. That way, the architecture has learned.

In utility learning, the procedual memory is used. The procedual memory contains productions rules and it is these that are retrieved when encountering a new problem. The probability of reaching the goal by choosing production rule P_i are given by $P_i = \frac{Successes}{Successes+failures}$. This means that every time a situation occurs, the number of $successes$ and $failures$ will change, and the architecture has then learned. (The value of P) will change due to the change in $successes$ and $failures$

> **Task 1.20 3p**: Describe how ACT-R retrieves relevant information in its cognition process.

Sander

Retrieval in ACR-R is based on different formulas. These formulas calculates the probability of success if this chunk is activated, or if this production rule are chosen. The formulas retrieves information based on how recently the chunk or production rule was used, and the correlation between the current situation and the information it has decided to retrieve.

Marius

The ACT-R has two primary buffers that hold information. The *manual buffer* is the buffer containing information about actions whilst the *visual buffer* contains information about locations and what kind of objects the system is exposed to.

The way these two are connected is by a pattern recognition function. If the system requests information needed to a cognition process, it will create a request where the system already has created some constraints. These constraints could be attribute values that restrict the search based on visual properties. An example of this could be the colour green. However, to identify the object, the system again has to send a request to the "what"-system. This will make the system change attention from where to what is it looking at.

Assignment 4

> **Task 1.1 2p**: What are some key characteristics of connectionist approaches?

Some key characteristics:

- The connectionism tries to explain the intellectual abilities.
- They are represented as an artificial neural network.
- It has some input-values that have a connection with hidden units that all have weights that are connected to some output units.
- The central goal in connectionist research is to find the right set of weights to accomplish a given task.

1.1 Artificial neural networks

> **Task 1.3 2p**: What is meant by the statement that artificial neural networks are universal function approximators?

Basically, it means that an artificial neural network can generate a class of functions that gives a valid output value, based on input value and training data.

> **Task 1.4 2p**: Explain, using also a formal description, why image classification can be addressed by a universal function approximator

Image classification can be addressed by a universal function approximator because of something called convolution. Input is an image, seen as RGB values, output is a classification of the image. The input values are fed to filter units, that detects certain features of the image, such as a dog's nose or elephant ear. After detecting all these features, the image are reduces to a *feature map* (which consists of all unique features found in the input values.). The feature map are then put into so called hidden layers, for further detection of features in the image. Following this, the output from the hidden layers are passed to a classification layer, where a label to the image is assigned. I.e "dog".

> **Task 1.6 3p**: What is the most important difference between an sigmoid activation function and the ReLU activation function?

The most important difference between a sigmoid activation function and a ReLU activation function is that a sigmoid activation function will map all all negative values close to 0 and all positive closer to 1 ($\frac{1}{1+e^{-x}}$), giving it a steep curve from x = 0 to x = 1, where the ReLU function maps all negative values to 0 and all positive to themselves ($f(x) = max(0, x)$), giving it a linear curve from x = 0 to x = 1.

> **Task 1.7 3p**: Explain the "association black box" analogy of artificial neural networks

The association black box analogy is that in a deep neural network, all computation happens in "a black box". We can not percieve the computation, only the input and output values.

1.2 Perceptron

> **Task 1.10 2p**: Give an example of a problem that can be solved by a perceptron. You might sketch this.

Classification of elephants and mice, based on their weigth can be solved by a perceptron. The elephants will be heavy, while the mice will be light. This way, they are linearly seperable in i graph, and a perceptron can solve the problem.

> **Task 1.11 3p**: Give an example of a problem that cannot be solved by a perceptron

XOR-operations cannot be solved by a perceptron, due to the fact that they are not linearly separable.

> **Task 1.12 3p**: Implement the perceptron model using python. Test your model with the AND and OR functions. Provide your code and explain your results

Code:

```
import numpy as np
import random

class Percepton:
    def __init__(self, learning_rate=0.01):
        self.learning_rate = learning_rate
        self.threshold = 1000
        self.weights = np.zeros(3)
        self.bias = random.uniform(-0.5, 0.5)

    def predict(self, inputs, train=True):
        yp = np.dot(inputs, self.weights)
        return 1 if yp > 0 else 0

    def train(self, train_data):
        for i in range(self.threshold):
            x, expected = random.choice(train_data)
            prediction = self.predict(x)
            self.weights += self.learning_rate * (expected - prediction) *
x
```

Results:

```
------------ AND ------------             ------------ OR ------------
[0, 0] --> 0                              [0, 0] --> 0
[0, 1] --> 0                              [0, 1] --> 1
[1, 0] --> 0                              [1, 0] --> 1
```

```
[1, 1] --> 1                            [1, 1] --> 1


Weights: [ 0.02  0.01 -0.02]            Weights: [0.01 0.01 0.   ]
Bias: -0.02                             Bias: 0.0
```

We see that for predicting the AND function, each input has to be 1, for the prediction to be 1. Having the weights $[0.02, 0.01]$ and bias -0.02, we need both inputs to be one, otherwise, the prediction will return 0 because the dot product would result in either $0.02 + 0 - 0.02 = 0$ or $0 + 0.01 - 0.02 = -0.01$. For the OR-function, it is the other way around. We need *at least* one input to be 1, or else the predict function will return 0: $0 + 0 - 0 = 0$.

1.3 General discussion

> **Task 1.16 2p**: Do ANN's use symbols? Explain how or how it is not possible.

ANN uses "distributed representations," Which means that a state is represented by activating a set of neurons. An idea is not stored in a single neuron; it has to be represented by several neurons. This is possible since the neural network is a connectionist.

> **Task 1.18 3p**: Is there something that coarsely resembles production rules in ANNs?

The input-specific weights of the different nodes in an ANN coarsely resembles production rules.

> **Task 1.19 3p**: Some connectionists argue that the brain's neural net indeed implements a symbolic processor. How do they argue for this claim?

Most connectionists claim that information is stored non-symbolically in the nodes' weigths, while others claim that the brain's net implements a symbolic processor. They think that yes, the brain is still a neural net, but also a symbolic processor at a higher and more abstract level. Following this, the connectionist research is to descover how the machinery for symbolic processing can be forged from a neural network.

2 Belief-Desire-Intention (BDI) architecture

> **Task 1.21 3p**: Explain the different architecture components in the BDI architecture.

The different attribute components in the BDI architecture is *Beliefs*, *Desires* and *Intentions*.

- Belief
  - Consists of information the agent has about the world. This information may or may not be true, it is what the agent *belives*.
- Desires
  - States of affairs that the agent would wish to be brought about
- Intentions
  - Desires that the agent has committed to achieving

> **Task 1.22 3p**: Comment on the view of 'practical reasoning' as referred to in the context of BDI?

Practical reasnoning is the process of finding out what to do. This comprises two activities; deliberaion and means-end reasoning. Deliberations means *what* state of affairs wr want to achieve and means-end reasoning means deciding *how* we want to achieve this state.

In a BDI, practical reasoning is done continously. The architechture runs in a loop that can be roughly summarized with the following loop:

```
while true:
    Observe the world
    Update internal workl model
    _Deliberate about what intention to achieve next_
    _Use means-end reasoning to get a plan for the intention_
    Execute plan
end while
```

This means that a BDI will not remember what it has previously done, only what is best to do and what it should do right now.

3 Subsumption

> **Task 1.23 3p**: What are the key ideas behind the subsumption architecture?

The key idea is to avoid having to use symbol manipulation to achieve human intelligence. Instead, it is aimed at real-time interaction and responses to a dynamic lab/office environment. It consisted of 4 key ideas:

- Situatedness - It should be able to react to its environment within a human-like time-frame. Perception of action.
- Embodiment - the agent should be embodied. This forces the designer to test and create a physical control system.
- intelligence - The architecture needs to be able to develop perceptual and mobility skills.
- Emergence - Individual modules are not considered intelligent by themselves. It is by looking at the agent and the environment that determines whether an agent is operating intelligence. (Determined by the observer.)

> **Task 1.25 2p**: Describe the layered structure of the Subsumption Architecture. How are the layers related to each other, how are their dependencies? How are the layers linked to sensor data in and actions out?

The subsumption architecture has a layered structure where lower, more primitive layers, has precedence over hoigher, more abstract, layers. Each layer is responsible for a singe behaviour goal, such as "avoid obstacle". Each layer is made up of connected, simple modules/processors that works as finite state machines. The layers communicate through the suppression of signals from other layers, which are communicated to an actuator. The actuator will co the output and percept new inputs.

4 Hybrid architectures

> **Task 1.26 3p**: What is the benefit of combining reactive and deliberative sub-components to create a hybrid architecture?

The benefit is that you "get the best of both worlds". You combine both perspectives in one arcitechture. The sub-components gathered from the reactive architecture will be able to respind to ral world changes without the need for ant complex reasoning and decision-making, and the deliberative sub-system will provide responsibility for abstract planning and decision-making using symbolic representations.