KLEX
Klusemann Extern
Marschallgasse 19-21
A-8020 Graz

AI

a categorization

Vorwissenschaftliche Arbeit
vorgelegt von Adriel Ondas

BetreuerIn: Manuel Menzinger

Schuljahr 2020-2022

Contents

1 What is AI?

1.1 Definition

First it is necessary defining what is meant by AI in this thesis, as there is a broad range of definitions.

To get an overview it is helpful to view an AI as an entity, a sort of black box. This construct can be called an "Agent". This thesis will shed some light into these black boxes.

1.1.1 Agents

This agent has to somehow get a sort of input, like sound from a microphone, or a click position on a website. Then, it has to process this input, and lastly it has to have some sort of output, like driving somewhere, or showing different things on a screen.

The input can be called "Percept", and the output "Action". Image

These Agents vary widely in their implementation and function, and can range from bots on the web, to the microchip in your smart stove or Roomba. Agents are this diverse because they have to act in different environments, for example a computer, a room where a robot is driving around, a map, a game, ...

This thesis however, will mostly focus on the internal structure of the agents, which can be similar, even in different environments. The goal of this thesis is an overview of these different internal structures.

As the concept of an agent is so openly defined, it helps to narrow it down further. For example a random number generator has all the prerequisites for an agent, but this thesis will focus only on intelligent agents.

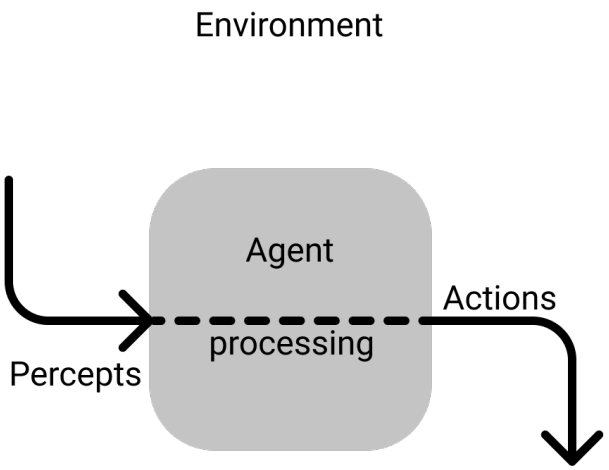

Figure 1.1: Interaction of an agent and its environment

1.1.2 Intelligence

For this, we must define intelligence first.

Commonly, intelligence is viewed as ”the ability to learn, understand and think logically about things; the ability to do this well”

Important to stress here are two separate concepts: Learning, and thinking logically. In AI, these concepts can be separated, as to say there is a learning, and an applied phase, where the logical thinking comes into play. There are also models combining these two phases, where learning takes place while also applying the previously learned.

1.2 Conclusion

This restriction of only intelligent agents is still not very narrow, as intelligent agents can still be a lot of things: A calculator has input, processes it intelligently, and shows the result, as does a player of a game, ...

1.3 General

1.3.1 Usage

There are very different models for AI, as will be explained in more detail in the next chapters, but of course they all have their own little problems but also advantages. So in practical use, different models are often chained together, to leverage the advantages of each sort of agent, to create an agent better than the individual agents.

1.3.2 Overfitting

When a model performs good on training, but bad on testing data, this is called overfitting. It happens when the model is ”memorizing” the answers, which results in less general knowledge.

1.3.3 Cross validation

To combat overfitting cross validation can be used. Cross validation is a process where the the border between training and test data is defined differently a few times, and the same learning algorithm is applied to all of these cases. Then the best one is used.

2 Categorization

2.1 Symbolic vs subsymbolic

A first big distinction is to make between symbolic and subsymbolic AI. The difference is in how knowledge about the environment is stored and used. In symbolic AI, knowledge is coded into an agent, while in subsymbolic AI knowledge is generally learned by an agent through observations (=data). These observations can either be pre-recorded or acquired by trial and error. (The structure is still hard coded)

Which sort of AI is better to use really depends on the circumstances: Symbolic AI for a complex subject normally requires more time to program in comparison, and needs a programmer well versed in the subject. Subsymbolic AI on the other hand depends on Data being available. Of course, it still is complex to program such a system, but the programmer does not have to know the subject at hand. Another aspect to keep in mind is the accuracy: Symbolic AI can have a 100 percent accuracy rate when programmed correctly, as it is just logical statements and numbers chained together by some operators, while with a subsymbolic AI it is normally hard to get a 100 percent accurate model, and most models are capped at some (high) percentage of accuracy. This is in part because of incorrect/incomplete data, but also because of considerations like time and resource constrains while training. Subsymbolic AI can still be more accurate in certain domains: When there is no one correct answer but good data for example. As it is often the case, there is a cost-accuracy payoff, and whether subsymbolic or symbolic AI is better suited really depends on the use case. Often they are combined to achieve better results. This work will focus on subsymbolic AI.

2.1.1 Data

The requirement for data in subsymbolic AI is also the reason why data collection is getting increasingly important in our time.

2.2 Supervised vs unsupervised learning

[1, p695] In supervised learning, in contrast to unsupervised learning labeled data is used instead of unlabeled data. Labeled data is easier to use, but harder to get. Labeled

data can be a lot of things, and it does not necessarily have to be human-labeled. A label can also be the output of an action, like e.g. the view time of a video on YouTube presented in different ways, or when the data is the weather yesterday, the label could be the weather today.

In supervised learning algorithms like backpropagation can be used to model a function, where some input-output pairs are given.

In unsupervised learning the AI has to find connections and the structure of the data on its own, as no clear input-output pairs are given. Here algorithms like support vector machines can be used to cluster data. Unsupervised learning can be used for e.g. serving ads based on interaction with an agent like Facebook. User data can be clustered and analyzed without the need for labeling.

Of course these approaches can also be combined: In so called semi-supervised learning only parts of the data are labeled. This approach brings with it the convenience of not having to label all the data, while still having the greater accuracy and ease of supervised learning.

2.3 Parameter(less)

[1, p737] In neural nets the most work is the decision of how many parameters are given to display a concept accurately. To eliminate this need there are also parameterless options. In many algorithms, like e.g. neural nets the biggest hurdle is finding a number of parameters big enough to accurately describe a concept, while still not overfitting.

The problem here is that this number of parameters is not universal, and has to be newly reconsidered for each problem.

So called parameterless approaches do not base their complexity on the problem, but rather on the training data available. Rather than representing the problem with parameters, often a part of the training data/the whole of it is used to come to a decision. This can be thought of as clustering. One example for a parameterless approach is K-nearest neighbors/Support vector machines section 3.3

3 Examples

3.1 Decision trees

3.1.1 Definition

A decision tree is a tree consisting of nodes, in which each node answers a yes/no question.

A decision tree can also be made symbolically, but to create such a tree subsymbolically, data is needed, which is used to shape the tree accordingly. The hard part is figuring out what question the next node will use for its split.

One method of measuring the usefulness of a question is the Gini method. It is calculated as follows: Gini impurity $= 1 - (yes/n)^2 - (no/n)^2$ The smaller this impurity, the less mixed the output is. Which is what we want, as less mixed means the tree is surer of its output. So to choose the next node to add to a tree, we measure the GINI impurity of all possible parameters, and choose the one with the smallest impurity. We repeat this process until we only have few samples left in each leaf.

Once created, to find an answer, one must walk down the tree, following the path the nodes lead you, until an end (a leaf) is met.

3.1.2 Usage

There are two advantages of decision trees in comparison to other forms of AI. One is the ease of understanding it, as we can retrace decisions with relative ease, which is very important in applications where trust in an algorithm is an issue. So for example in an environment where job applications or prison sentences are guided by algorithms, it might be required to be able to check the algorithm for, for example, discriminatory behavior.

Another benefit decision trees offer is the low computational complexity, in both training and application. Of course, they can grow arbitrarily large, but with the right techniques they can be brought back to reasonable sizes, which just capture the essentials.

There are two methods for cutting the size of a tree: Early stopping and pruning. In early stopping we stop a branch as soon as we cannot split the data with sufficiently low

Gini impurity. This can be very efficient, but there are cases where one question does not help, but two do. So to avoid stopping too early, one can first generate the whole tree, and then prune useless branches away.

Of course, decision trees are not perfect: For one they require a lot of data to be conclusive, and as they are one of the simpler forms of AI, they need to get quite big to capture complex concepts.

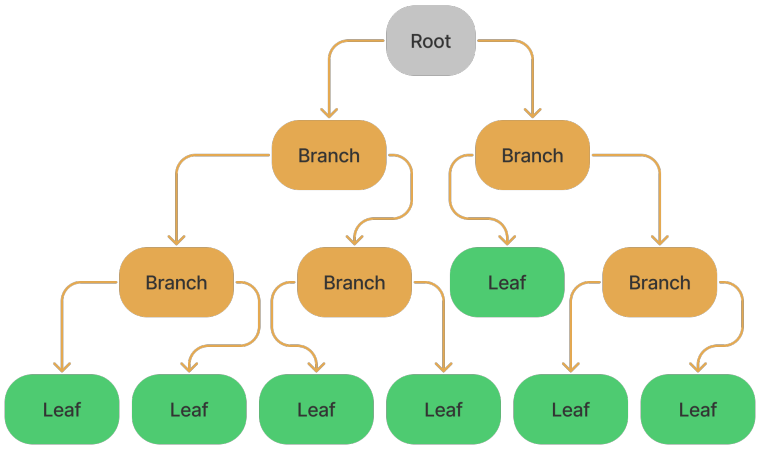

Figure 3.1: decision tree

3.2 Neural nets

3.2.1 Definition

[1, p727] A neural net is a structure where values are passed on through layers of nodes. Each node performs only a relatively simple operation with parameters inherit to each specific node.

First the amount of layers and nodes has to be defined, and to train the neural net, these parameters can be tuned.

As there are normally too many possible values for the parameters not every possible combination can be brute forced. There are different methods for tuning the parameters. Of course these parameters could, in theory, also be filled in by hand, but this is normally not feasible. Two methods to fill them in are back propagation and evolutionary learning.

In backpropagation 3.2, a supervised method, an input is given, and the output is compared to the known correct output. Now the nodes responsible for the biggest error are slightly nudged in the right direction.

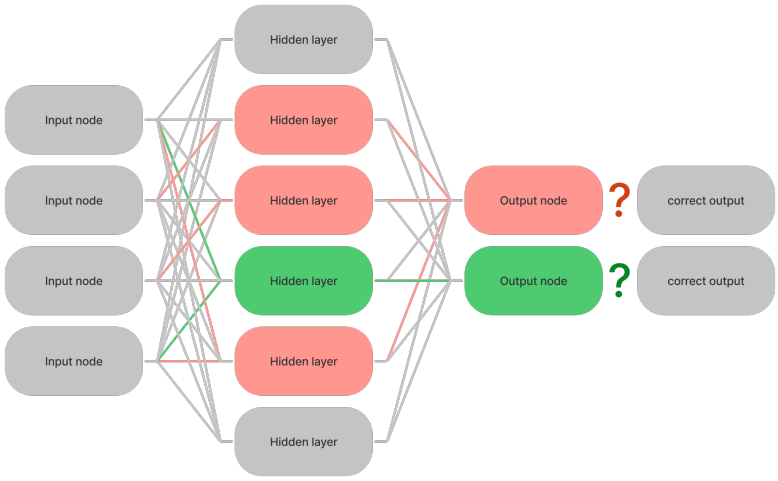

Figure 3.2: Neural network backpropagation

3.3 Support vector machines

3.3.1 K-nearest-Neighbours

As a parameterless approach, KNN works by saving the whole of the training data, representing it in a n-dimensional space, and letting the k-nearest neighbors decide how to act. There are only few parameters to control now, and the amount of parameters is no longer strictly bound by the complexity of the problem at hand. Mainly, there remain: the k, the weight of the different dimensions, and the way the neighbors are combined. The weight depends on the real-live weight of the dimensions (e.g. color could have a lower significance than the size of an object to categorize). As for the ways in which the neighbors are combined: There are many options here, like e.g. Majority vote, average, weighed average, ...

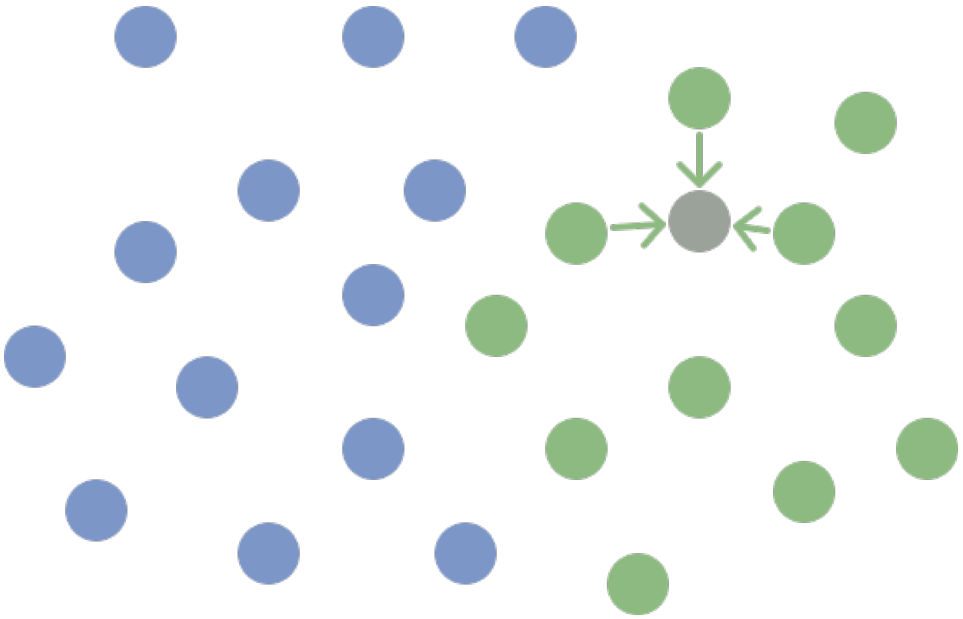

Figure 3.3: K-Nearest-Neighbours

3.3.2 SVMs

A problem with KNN type approaches is high memory usage, as all the training data has to be stored. To combat this, one can use the fact that in most problems the entries close to the border are more helpful than entries in the middle of a decisive cluster.

So as an extension of KNNs, SVMs were developed. They come with most of the strengths of KNNs, while eliminating some weaknesses.

In a SVM only the few entries along the border are stored, these are called support vectors, as they ”hold up” the border. When a decision needs to be made, only the side of the border has to be checked to come to a conclusion. This can also mean a significant increase in efficiency, as not all distances have to be checked like in KNN.

This reduces the problem to one of fitting a curve to some points, which is already well-researched, and the parameters to choose here are on how detailed the curve can be while still not overfitting.

3.3.3 Usage

SVMs are often the first approach tried, as they are relatively simple, and yield good results in a wide variety of use cases.

They are relatively simple to view and understand, should there only be few important dimensions.

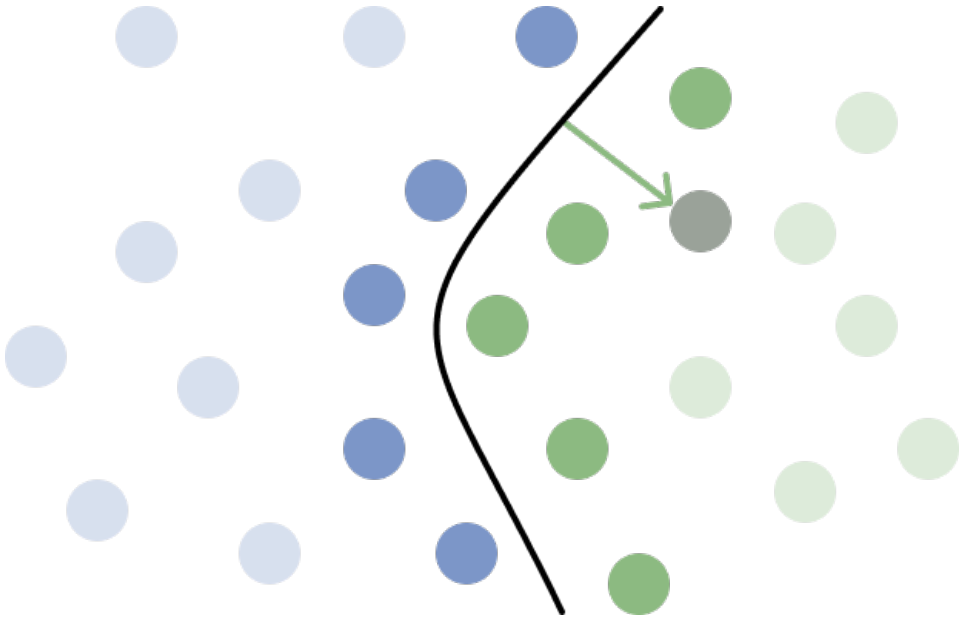

Figure 3.4: Support Vector Machine

3.4 Comparison

complexity: SVM/DT/NN accuracy: depends training: SVM pro application: performance resources

List of Figures

Bibliography

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Pearson Education, Inc., 3 edition, 2010.