Chalmers Tekniska Högskola

DESIGN OF AI SYSTEMS (DAT410)

Module 1: AI problem solving

Group 38

Himanshu Sahni
9812075498
sahni@chalmers.se
MPCAS

Nina Uljanić
9609134920
uljanic@chalmers.se
MPCAS

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part in other solutions.

Problem 1: Predict the temperature

Problem Formulation:

1. For the same weekend
2. For the same date as today but next year

Problem Classification:

Regression

Problem Characteristics:

This problem has a set of different parameters which need to be considered to develop a solution.

1. We have two different time periods for which we need to predict the temperature. One is a short period and the other one is a long period. We need to measure the parameters at different time periods throughout, some parameter needs to be collected more frequently than others.
2. We need to take into account the meteorological variables for that location into account, for example, wind speed, temperature, humidity, wind direction, atmospheric pressure, and general weather over years.
3. We need to consider the topological parameters as well, for example, height above sea level, and soil properties of the location.

Approach:

This problem can be tackled in the following way:

1. Data collection - The collection of data with all the parameters mentioned above over years is needed to develop a model.
2. Data analysis - Analyze the data before feeding it into the model, usually known as preprocessing and consists of scaling, one hot encoding, etc.
3. Developing the model
4. Training the model
5. Numerical prediction of parameters (testing)
6. Post-processing Analysis

Possible Method:

Our understanding of the problem is that we need a lot of data to solve this problem and it can be considered as a problem that comes under supervised learning since we have labeled data available.

This is the problem of regression since we have to predict the value at a particular time. One of the possible and most recommended way of solving this problem is to use neural networks.

Now, here we need to calculate only the temperature value and we have enough parameters to develop a formula with which this can be done. The neural network, consisting of input neurons, hidden layers, and an output layer, can be fed the aforementioned parameters. The input layer would be consisting of input neurons, one for each of the parameters in the data set. We would try out a different number of hidden layers and hidden neurons and compare the results for the combinations.

Once these parameters are fixed, the network weights and threshold can be trained and set to minimize the prediction error. The best approach to perform this would be using backpropagation. Then we'd consider the parameters for the networks i.e., the loss function, activation functions, and optimizers. In general, it is observed that the results are best with the MSE (mean squared error) loss function, but the other options can be tried as well in order to find the best fit. The ReLU activation function is commonly used for the hidden layers, and for the output layer we can use a sigmoid or a linear function, depending on the desired format of the output. An important point to note here is that we need only one unit for the output layer which should tell the temperature value. We have different options for optimizers as well as link ADAM, RMS prop, and SGD. It is not possible to make a comment on which one would work the best in this scenario, but the different options can be tried out.

Problem 2: Bingo lottery problem

Problem Formulation:

In a bingo lottery, 1000 lottery tickets are printed. Every ticket consists of a 5x5-square where each of the numbers 1 to 25 are randomly placed. A winning combination is received for a horizontal, vertical, or diagonal row. For the draw of the numbers, it is desired to select a set of numbers so that exactly 120 tickets give a win (or at least as close as possible), and no ticket must give more than one win. Knowledge of all the lottery tickets can be assumed.

Problem Classification:

Constraint satisfaction problem

Problem Characteristics:

Relevant characteristics of the problem:

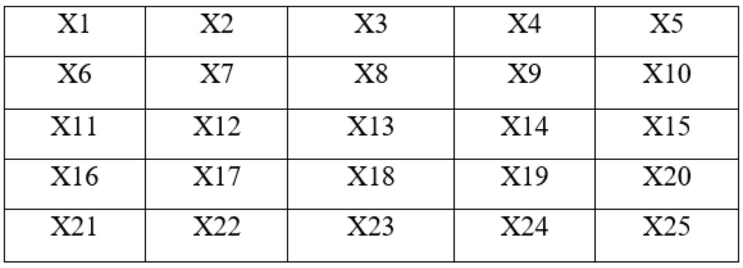
- 1000 tickets of which 880 were without prized and 120 with the prize.
- Ticket consist of a 5x5 square grid.
- Numbers 1 to 25 are randomly placed on each ticket.
- Winning condition is either horizontal, vertical, or diagonal row consisting of a specific set of numbers (the order is irrelevant).
- One ticket cannot give more than 1 win.
- Tickets cannot be equal (no repetition).

Possible Method:

Using a Factor graph with the above-defined constraints

Approach:

Figure 1 shows a theoretical sample ticket, consisting of 25 numbers (range 1 to 25) placed randomly in each of the 25 squares. These are the variables for our factor graph. Note: just for the sake of explanation we have written the numbers in sequence here but they can be completely random.



X1	X2	X3	X4	X5
X6	X7	X8	X9	X10
X11	X12	X13	X14	X15
X16	X17	X18	X19	X20
X21	X22	X23	X24	X25

Figure 1:

We need to create factors with all the constraints satisfied and we can do it in the following way: The figure shows the factors in the case of a ticket containing the winning numbers in a row. There are 5 possibilities of factors since we have 5 rows in a ticket (one factor per row). F1 denotes the factor for row 1 and similarly, we have F2, F3, F4, and F5 for the consecutive rows. The winning chance of a ticket is dependent on whether the value of the factor is 1 or 0, where 1 denotes that it has the 5 winning numbers(the sequence is not important) and 0 denotes that it does not.

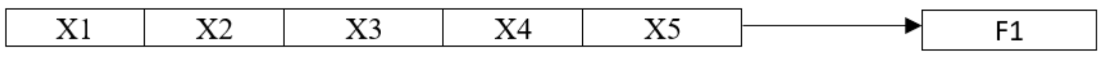


Figure 2:

A similar approach is applicable in the case of the columns as well. There are 5 possibilities of factors since there are 5 columns in a ticket (one factor per column). F6 denotes the factor for column 1 and similarly, we have F7, F8, F9, and F10. The winning chance of a ticket is dependent on whether the value of the factor is 1 or 0, where 1 denotes that it has the 5 winning numbers and 0 denotes that it does not.

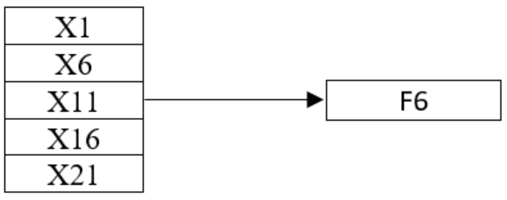


Figure 3:

In the case of diagonal, we will have only 2 factors since there are only 2 diagonals in a ticket. So the possible factors are F11 and F12.

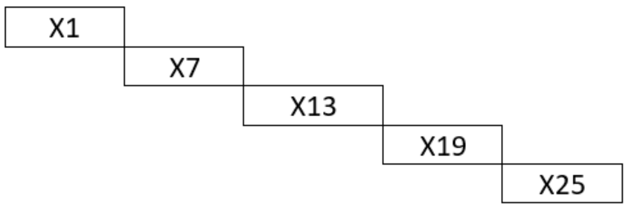


Figure 4:

We solve the problem using the backtracking algorithm, where we scan the numbers from left to right for every row, top to bottom for every column, and across the diagonals. If we find a number out of the 5 winning numbers we continue the search in that direction or else we backtrack. Here we assume that we will be creating the winning tickets first and since it is mentioned that there should be 120 winning tickets, we divide them into 40 tickets having the winning numbers in rows, 40 in columns, and 40 in diagonals.

A row winning ticket will have the 5 winning numbers in a random row, placed randomly. It is not possible to have them in any column or diagonal combination at the same time. A column winning ticket will have the 5 winning numbers in a random column, placed randomly. It is not possible to have them in any row or diagonal combination at the same time. A diagonal winning ticket will have the 5 winning numbers in a random diagonal, placed randomly. It is not possible to have them in any row or column combination at the same time. It should be noted here that we will have 3 different categories of factors since we have 3 different conditions for the winning tickets. The 3 different categories are represented below:

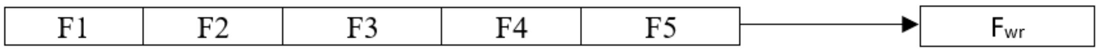


Figure 5:

The first set of factors is from the rows. F1 represents row 1 if it has the winning numbers. The same goes for others. A winning ticket can have only 1 of the factors as true i.e., value 1, and the others as 0. For example, if we have the winning number in row 3 then the value for F3 = 0 and all other factors are 0. Therefore, F1 + F2 + F3 + F4 + F5 = 0 + 0 + 1 + 0 + 0 F_{wr} = 1 (the only possible values are 1 or 0) F_{wr} represents the winning row factor (Figure 5).

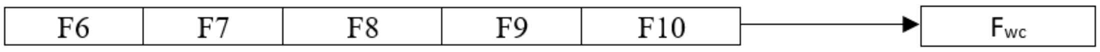


Figure 6:

Similarly, for the columns, F_{wc} = F6 + F7 + F8 + F9 + F10 (the only possible values are 1 or 0) F_{wc} represents the winning column factor (Figure 6).

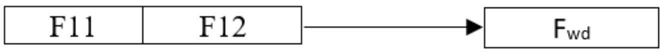


Figure 7:

Similarly, for the diagonals F_{wd} = F11 + F12 (the only possible values are 1 or 0) F_{wd} represents the winning diagonal factor (Figure 7).

Finally, we do a weighted sum of the winning row, column, and diagonal factors i.e., weight = F_{wr} + F_{wc} + F_{wc} and evaluate the objective function. Some of the important points that we need to take care of while implementing the algorithm are:

- We need to iterate until we receive the required number of solutions i.e., 120 tickets.
- We need to store the winning combination such that there is no repetition of the solution

For the non-winning tickets, we just need to define a factor such that it does not fulfill the above constraints i.e., the 5 winning numbers are not present in either a row, column, or diagonal.

Problem 3: Public transport departure forecast

Problem Formulation:

At the stops for public transport, there is often a screen predicting the next departure time for each line. Suggest different approaches for how this time can be estimated.

Problem Classification:

Prescriptive Modeling (Prediction)

Problem Characteristics:

To predict the time accurately we need to take into the following scenarios:

1. Near real-time location of the vehicle.
2. The traffic congestion (density) on the route at different time intervals of the day, and on different days of the week.
3. The weather conditions.
4. Detailed information about the routes, distances between the stops, number of stops on the route, and total distance to be covered on the route.

There is another factor that should be taken into account i.e., the passenger flow, which might affect the boarding time for the stops. This factor can be considered as time spent on a stop. However, we feel that this might not be a very major factor and might not have a significant effect when compared to the others.

Possible Method:

After reading through a few research papers, it was found that most common methods used include:

1. Artificial neural Methods/ Support Vector machines
2. Kalman filter-based models.
3. Regression models like K nearest neighbors
4. Long Short-Term Memory

Approach:

First, we consider the requirements of such a system. We do not think that the real-time location of the vehicle is crucial to the problem; in fact, periodic location updates would be sufficient (with the frequency of updates assigned based on the preference of the stakeholders). The public transport departure forecast does not benefit from the constant calculation, it only wastes resources unnecessarily. Following the principles of sustainable computing, the number of calculations performed should be at a minimum, such that the requirements of the system are satisfied.

1. The first task should be to collect data. We need to establish the data sources for traffic and weather, which collects and stores the data over long periods of time.

2. A database needs to be implemented that stores the vehicle's movement and the traffic flow of the route taken. The database should have enough data with a high update rate to calculate the distance, speed, and the effect of other parameters mentioned so as to predict the time accurately.

3. Selection of parameters that are to be used as the inputs in the model, for example, traffic congestion, and the weather should be given as inputs to the network while training.

4. Perform training and testing and evaluation of the model. The need for the model is to predict the time in advance, this means the network needs to learn the trends of the flow of traffic under different conditions and time periods. We can relate to recurrent neural networks, for example, the system learns the writing style of an individual and then autocompletes the sentence and even give some suggestions. This is how LSTMs work.

The method implemented should be able to calculate the arrival time of the bus at the following bus stops based on the current location of the vehicle, the predefined schedule of the bus and other factors such as traffic congestion and weather, among others. It is assumed that a bus shall wait until the scheduled departure if it arrives early at the bus stop, however, this is not something we can enforce, and therefore, our method should be able to handle exceptions. This should not be a problem, however, as we'd have periodic location updates on which the system would base its calculations. The method should also take into account the travel times based on the given time period during the day. One of the ways of doing this can be by gathering information like speed from the buses or vehicles that have traveled before on the same route. The accuracy of this model is highly dependent on the update rate of the parameters in the model as it will affect the calculation of time directly.

Problem 4: Film festival problem

Problem Formulation:

Quite a few years ago we had a system at the CSE department creating automatic schedules for film festival enthusiasts, enabling everyone to quickly order tickets before they run out. When the festival programme arrives, all films are quickly entered. Individuals can then indicate desired films with a priority and the system automatically creates an entire schedule for the week. The challenge is that it is usually impossible to see all films since many are shown at the same time, and each film is often shown several times during the week.

Problem Classification:

Constraint optimization (schedule) problem

Problem Characteristics:

The aim is to create a schedule such that the maximum number of people gets to watch the movies as per their preference list (order is important). A movie screening is considered an event happening at a fixed time and they can be treated as blocks if the timings overlap. If two movies are played at the same time, that is, the air times overlap, the attendee should be assigned the slot for the movie with higher priority. Any skipped movie could potentially be watched at a later time, provided its air time is not in conflict with another movie placed higher on the priority list. In short, this scenario can be formulated as an optimization problem:
Minimize "dissatisfaction" subject to

- the number of tickets are limited for every movie screening
- all attendees get assigned the movies in order of expressed preference
- only one movie can be assigned to a time slot
- all attendees get to watch a movie once

Possible Method:

Constraint satisfaction problem

Approach:

Here, the time slots can be treated as variables and the movies (as per the priority of the attendee) as the constraints. The model should be based on the CSP backtracking algorithm (see problem 6) where we loop through all the possible solutions.

We define an objective function and we look for all the possible solutions to it, however, it is important to know that the solution found might be suitable but not the most optimum one. Therefore an optimal solution to the objective function will be the one with the highest possible weight, calculated using the constraints and the variables. Additionally, once we have the optimal solution based on the constraints we can seek validation from the user if it suits them.

Problem 5: Product rating in consumer test

Problem Formulation:

For E.g. how to determine the score for different dishwashers in a consumer test?

Problem Classification:

Function regression problem

Problem Characteristics:

Here we assume that the data collected in a consumer test includes ratings of various characteristics of the dishwashers, as well as a textual review. The scores for the dishwashers are therefore based on the ratings by the consumers for the different parameters mentioned below and text feedback.

The appliance's rating can be evaluated by the following characteristics:

- data set of its performance, for example, electricity, water, and cleaning agent consumption
- cost of the appliance
- product usability
- product design
- product features
- customer review (qualitative and quantitative).

Possible Method:

Regression

Approach:

As mentioned, the score for a dishwasher includes both the ratings on the parameters and the text feedback. When the review is given as text only, a machine learning algorithm paired with natural language processing (NLP) is used to calculate the score. The system needs to be able to take in the texts (reviews) which are analyzed using NLP techniques. The input text must be preprocessed, which includes, but is not limited to, removing punctuations and stop words.

The machine learning model must then be trained to determine the score; this is done through supervised learning. Labeled data (text with accompanying score) is fed into the machine learning model and the weights are adjusted until the model successfully classifies input text with a matching score.

The final score is calculated using a function in which the variables are the constraints mentioned above. We evaluate the function value using the above-calculated score from the text and the numerical ratings from the consumers.

Problem 6: Constraint satisfaction and constraint programming

Constraint programming

Constraint programming is used for finding solutions to different combinatorial search problems in different disciplines like algorithms, operations research, etc. It is a form of programming in which mathematical constraints define the relations between the variables in a program. In simple terms, the main idea is that the user provides a set of constraints to be solved using a general-purpose constraint solver.

Constraints can be defined as the relations between different variables and can be of different types. For example, arithmetic constraints are defined using arithmetic operators like $<$, $>$, $=$, $\geq$, $\leq$. Logical constraints are restrictions with clear meaning imposing a condition,for example "at most." Extensional constraints consist of a set of values for which the condition will be satisfied.

The aim of constraint programming is to find feasible solutions from a given set, and the problem can be modeled in terms of arbitrary constants. One of the most used scenarios for constraint programming is in finding solutions to scheduling and combinatorial optimization problems

Constraint satisfaction problems

Constraint satisfaction problems (CSPs) can be defined as a set of mathematical problems that consists of objects whose state must comply with the given constraints (restrictions). These problems consist of a set of variables that have a defined domain and a set of restrictions on the values that the variables can take. A general structure of a CSP is as follows:

- A CSP consists of three components (V, D, C) where, V is a set of variables {v1, v2, v3,...}, D is a set of domains {d1, d2, d3, ...} (one for each variable), and C is a set of constraints that specify the allowable combination of values.
- A constraint C is a pair (S, R) where, S = $(x_{i1}, ..., x_{ik})$ are the variables of C (the scope, defined as a set of variables that participate in the constraint), and R $\subseteq$d$_{i1}$...d_{ik} are the tuples satisfying C (relation, defined as the values that the variable can take).
- The solution of a CSP are the assignments that satisfy all the constraints.
- In a CSP the domain can be of two types:
  1. Continuous - the phases have a finite domain for one variable and can describe only one area. It is also referred to as constant areas.
  2. Discrete - it is possible to have a single state with many different variables in this endless space. Every parameter, for instance, may get an infinite number of initial states.

- There are three types of constraints possible:
    1. Unary - these restrictions can only limit the value of one variable.
    2. Binary - these restrictions involve two variables. A value between x1 and x3 can be found in a variable named x2.
    3. Global - this kind of restriction includes an unrestricted amount of variables.

Solving a Constraint satisfaction problem

One of the most common methods to solve the CSPs is using the backtracking algorithm. The backtracking algorithm has a depth-for-search approach in which we choose the values for a variable at a time and then backtrack until left with no legal values to assign. The aim is to seek a path and reach the desired solution by passing through some checkpoints in between. During traversing, if the checkpoints don't lead to the right solution, we can return back to the preceding checkpoint and explore a different path. The backtracking algorithm is also referred to as a brute-force algorithm technique. This algorithm involves multiple steps and they are described below:

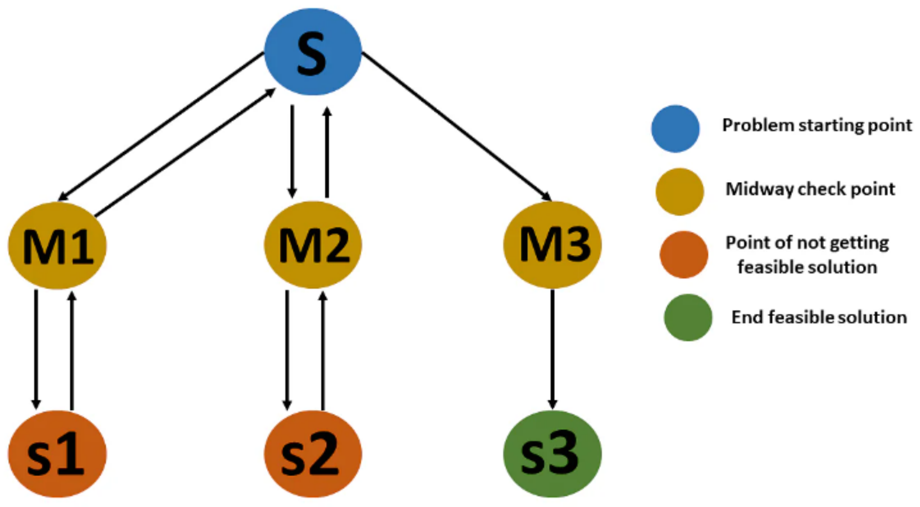


Figure 8: Source: https://www.simplilearn.com/tutorials/data-structure-tutorial/backtracking-algorithm

- Consider the example shown in Figure 8. The graphical representation is known as a space state tree which represents all the possible states in a given problem.
- S represents the starting point; M1, M2, and M3 represent the midway (intermediate) points; S1, S2, and S3 represent the possible solutions among which S3 is the desired solution.
- We start from point S and move towards the solution S1 traversing through point M1. However, since S1 is not a viable solution, we backtrack first to M1 and then back to S, from where we explore the next path, M2. In this way, we reach the desired solution.
- It is clear from the figure that S3 is the desired solution. It is reached after traversing through all the preceding combinations.

- The algorithm returns a success if the current point is the desired solution. Otherwise, if we don't find a feasible solution after all the paths have been exhausted, it returns a failure.
- These steps are repeated and the backtracking continues until we reach the desired solution. Backtracking algorithm can be used to solve a variety of problems. Some of the examples include decision-making problems, optimization problems, etc.

Individual Summary and Reflection

Lecture 1: Design of AI systems - introduction (2023-01-17)

Lecture summary by Himanshu Sahni

- History of AI: We looked into the history of AI and how it started, for example, the Turing machine, its computation, input, memory, and output. The Turing test is the definition of intelligent i.e., self-recursive if people think it's intelligent, it is intelligent. Some of the practices in AI solved problems that are reserved for humans and non-numerical human insight. We noticed that it is programmed with a set of rules and logic which does not work for a lot of problems and it is actually difficult to write abstract rules.
- AI computation and Statistics: Looked into basic computing consisting of word processors, networks, and spreadsheets. Advanced computing, is a mix of human insight and data. For example, simulations, signal processing., algorithms, optimization, and operations research (flight scheduling).
- Data science: it was founded on statistical methods and we investigate to understand and predict. AI systems are systems that perform intelligent tasks. Some examples of AI methods are problem-solving (search), knowledge planning, uncertain knowledge, reasoning, learning (inference), communicating, and perceiving (image recognition).
- Machine Learning: self-improving process with less insight and more data.
- Modeling: Image learning, computer vision, learning patterns, vectors, and shapes.

Lecture summary by Nina Uljanić

The first lecture introduced the history of artificial intelligence (AI). To understand the beginnings of AI, one must learn about the mathematician Alan Turing and his two creations, the Turing machine - an abstract machine able to perform any computation, and the Turing test, which asks what is intelligence and suggests that whether an entity is intelligent is something that can be agreed on.

Previously, it was thought that so called "intelligent" tasks or problems could be used to test intelligence. An example of such a problem is the game Chess. The field of AI has developed far enough to prove this assumption wrong. There now exist AI which successfully outperforms a human player. There exists, however, a plethora of tasks in which AI doesn't outperform a human, such as identifying a dog. The reason is that such problems are difficult to define in terms of rules.

The most important part of the lecture was, in my opinion, the list of various applications of AI which are "hot" - natural language processing, image classification and video analysis, anomaly detection, diagnostics, logistics, etc. The quote "data is the new oil" was also of value, making us consider the ethical aspects of the field.