Now my son Siddharth’s age is 4 years and we are working parents. We always wanted to take care of Siddharth without the help of daycare or creche. With this constraint, anyone has to be available at home to take care. When I used to be there at home, I had a big confusion to understand why Siddharth cries. The reasons could be feeling hungry, feeling thirsty, feeling irritation due to cloths or due to anything etc. I think it's very common to any parents.”

***My question was always, is there any device which can help me to understand the reason behind his tears.***

Assume that I have the following device at home and it receives baby crying signals and classifies them. It alarms with different music when a baby cries for a certain reason like feeling hungry, feeling thirsty, feeling irritation due to clothes, etc.



To design such a device, the fundamental knowledge required is a huge understanding of babies' emotional behavior. This knowledge becomes an experience. Using this experience, generalized rules can be defined. For example, a baby would cry after a sleep due to hunger, a baby would cry during sleep due to irritation, etc. So these generalized rules will become the model. This model can be tested for different babies and model correction can be made to get better accuracy.

The final model is embedded on the device. The device accepts the signals, extract the features of the signal and input the features of the signal to the model to classify the emotion of the baby.

In this short imagination story, three elements are important to us. They are — Experience, what is the task or objective of the problem to solve and performance measure to evaluate how well the model is predicting.

***Definition of Machine Learning (Mitchell 1997) —****“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at the tasks improves with the experiences”*

I hope now you are able to map the experience E, Class of Tasks T and Performance measure P with the imagination story we discussed above.

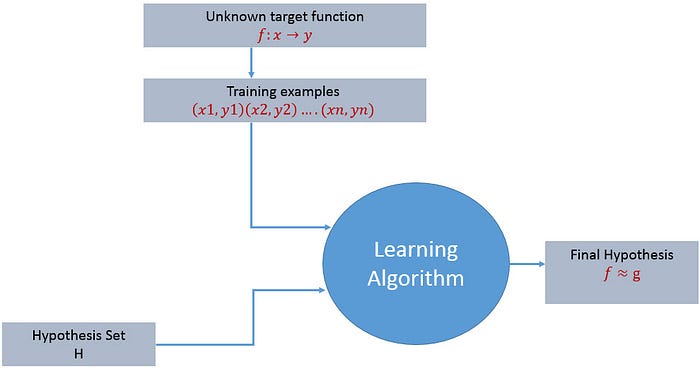
Assume we want to classify an incoming mail as a Spam or Not. This problem can be described in terms of three elements as below :

**Spam Mail detection learning problem**

1. **Task T:** To recognize and classify emails into ‘spam’ or ‘not spam’.
2. **Performance measure P:** Total percent of mails being correctly classified as ‘spam’ (or ‘not spam’ ) by the program.
3. **Training experience E:** A set of mails with given labels (‘spam’ / ‘not spam’).

**2. Simple learning process**

For any learning system, we must be knowing the three elements — **T (Task)**, **P (Performance Measure)**, and **E (Training Experience)**. At a high level, the process of learning system looks as below.



The learning process starts with task T, performance measure P and training experience E and objective are to find an unknown target function. The target function is an exact knowledge to be learned from the training experience and its unknown. For example, in a case of credit approval, the learning system will have customer application records as experience and task would be to classify whether the given customer application is eligible for a loan. So in this case, the training examples can be represented as (x1,y1)(x2,y2)..(xn,yn) where X represents customer application details and y represents the status of credit approval.

**With these details, what is that exact knowledge to be learned from the training experience?**

So the target function to be learned in the credit approval learning system is a mapping function f:X →y. This function represents the exact knowledge defining the relationship between input variable X and output variable y.

Next, the learning algorithms try to guess a “hypothesis’’ function h(X) that approximates the unknown f(.). A hypothesis is a function that best describes the target and Hypothesis set or space H(.) is the collection of all the possible legal hypothesis. This is the set from which the machine learning algorithm would determine the best possible (only one) which would best describe the target function or the outputs. **The goal of the learning process is to find the final hypothesis that best approximates the unknown target function.**

**3. Design of a learning system**

Just now we looked into the learning process and also understood the goal of the learning. When we want to design a learning system that follows the learning process, we need to consider a few design choices. The design choices will be to decide the following key components:

**1. Type of training experience  
2. The exact type of knowledge to be learned (Choosing the Target Function). Initially, the target function will be unknown.  
3. A representation for this target knowledge (Choosing a representation for the Target Function)  
4. A learning mechanism (Choosing an approximation algorithm for the Target Function)**

We will look into the checkers learning problem and apply the above design choices. For a checkers learning problem, the three elements will be,

*1. Task T: To play checkers  
2. Performance measure P: Total percent of the game won in the tournament.  
3. Training experience E: A set of games played against itself*

**Training experience**

During the design of the checker's learning system, the type of training experience available for a learning system will have a significant effect on the success or failure of the learning.

1. **Direct or Indirect training experience —**In the case of direct training experience, an individual board states and correct move for each board state are given.  
   In case of indirect training experience, the move sequences for a game and the final result (win, loss or draw) are given for a number of games. How to assign credit or blame to individual moves is the credit assignment problem.
2. **Teacher or Not —**Supervised — The training experience will be labeled, which means, all the board states will be labeled with the correct move. So the learning takes place in the presence of a supervisor or a teacher.  
   Unsupervised — The training experience will be unlabeled, which means, all the board states will not have the moves. So the learner generates random games and plays against itself with no supervision or teacher involvement.  
   Semi-supervised — Learner generates game states and asks the teacher for help in finding the correct move if the board state is confusing.
3. **Is the training experience good —**Do the training examples represent the distribution of examples over which the final system performance will be measured?  
   Performance is best when training examples and test examples are from the same/a similar distribution.

The checker player learns by playing against oneself. Its experience is indirect. It may not encounter moves that are common in human expert play. Once the proper training experience is available, the next design step will be choosing the Target Function.

**Choosing the Target Function**

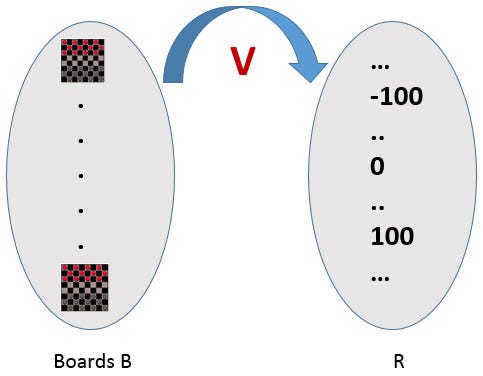
In this design step, we need to determine exactly what type of knowledge has to be learned and it's used by the performance program.

*When you are playing the checkers game, at any moment of time, you make a decision on choosing the best move from different possibilities. You think and apply the learning that you have gained from the experience. Here the learning is, for a specific board, you move a checker such that your board state tends towards the winning situation. Now the same learning has to be defined in terms of the target function.*

Here there are 2 considerations — direct and indirect experience.

**During the direct experience**, the checkers learning system, it needs only to learn how to choose the best move among some large search space. We need to find a target function that will help us choose the best move among alternatives. Let us call this function ChooseMove and use the notation **ChooseMove : B →M** to indicate that this function accepts as input any board from the set of legal board states B and produces as output some move from the set of legal moves M.

**When there is an indirect experience**, it becomes difficult to learn such function. How about assigning a real score to the board state. So the function be **V : B →R** indicating that this accepts as input any board from the set of legal board states B and produces an output a real score. This function assigns the higher scores to better board states.



If the system can successfully learn such a target function V, then it can easily use it to select the best move from any board position.

Let us therefore define the target value V(b) for an arbitrary board state b in B, as follows:  
1. if b is a final board state that is won, then V(b) = 100  
2. if b is a final board state that is lost, then V(b) = -100  
3. if b is a final board state that is drawn, then V(b) = 0  
**4. if b is a not a final state in the game, then V (b) = V (b’), where b’ is the best final board state that can be achieved starting from b and playing optimally until the end of the game.**

The (4) is a recursive definition and to determine the value of V(b) for a particular board state, it performs the search ahead for the optimal line of play, all the way to the end of the game. So this definition is not efficiently computable by our checkers playing program, we say that it is a nonoperational definition.  
**The goal of learning, in this case, is to discover an operational description of V ; that is, a description that can be used by the checkers-playing program to evaluate states and select moves within realistic time bounds.**

It may be very difficult in general to learn such an operational form of V perfectly. We expect learning algorithms to acquire only some approximation to the target function ^V.

**Choosing a representation for the Target Function**

Now its time to choose a representation that the learning program will use to describe the function ^V that it will learn. The representation of ^V can be as follows.

1. A table specifying values for each possible board state?
2. collection of rules?
3. neural network?
4. a polynomial function of board features?
5. …

To keep the discussion simple, let us choose a simple representation for any given board state, the function ^V will be calculated as a linear combination of the following board features:

* x1(b) — number of black pieces on board b
* x2(b) — number of red pieces on b
* x3(b) — number of black kings on b
* x4(b) — number of red kings on b
* x5(b) — number of red pieces threatened by black (i.e., which can be taken on black’s next turn)
* x6(b) — number of black pieces threatened by red

^V = w0 + w1 · x1(b) + w2 · x2(b) + w3 · x3(b) + w4 · x4(b) +w5 · x5(b) + w6 · x6(b)

Where w0 through w6 are numerical coefficients or weights to be obtained by a learning algorithm. Weights w1 to w6 will determine the relative importance of different board features.

**Specification of the Machine Learning Problem at this time —**Till now we worked on choosing the type of training experience, choosing the target function and its representation. The checkers learning task can be summarized as below.

* **Task T : Play Checkers**
* **Performance Measure : % of games won in world tournament**
* **Training Experience E : opportunity to play against itself**
* **Target Function : V : Board → R**
* **Target Function Representation : ^V = w0 + w1 · x1(b) + w2 · x2(b) + w3 · x3(b) + w4 · x4(b) +w5 · x5(b) + w6 · x6(b)**

The first three items above correspond to the specification of the learning task,whereas the final two items constitute design choices for the implementation of the learning program.

**Choosing an approximation algorithm for the Target Function**

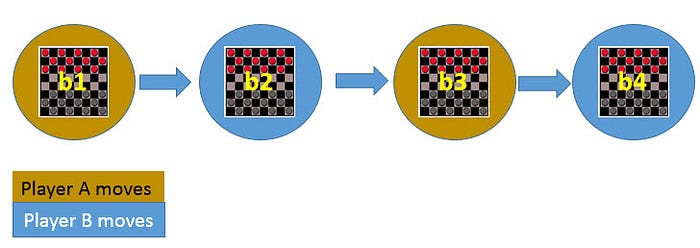
**Generating training data —**

To train our learning program, we need a set of training data, each describing a specific board state b and the training value V\_train (b) for b. Each training example is an ordered pair <b,V\_train(b)>

For example, a training example may be <(x1 = 3, x2 = 0, x3 = 1, x4 = 0, x5 = 0, x6 = 0), +100">. This is an example where black has won the game since x2 = 0 or red has no remaining pieces. However, such clean values of V\_train (b) can be obtained only for board value b that are clear win, loss or draw.

In above case, assigning a training value V\_train(b) for the specific boards b that are clean win, loss or draw is direct as they are direct training experience. But in the case of indirect training experience, assigning a training value V\_train(b) for the intermediate boards is difficult. In such case, the training values are updated using temporal difference learning. **Temporal difference (TD) learning is a concept central to reinforcement learning, in which learning happens through the iterative correction of your estimated returns towards a more accurate target return.**

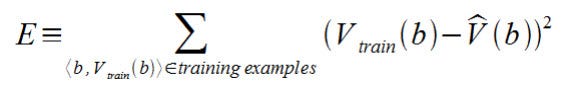
Let Successor(b) denotes the next board state following b for which it is again the program’s turn to move. ^V is the learner’s current approximation to V. Using these information, assign the training value of V\_train(b) for any intermediate board state b as below :  
**V\_train(b) ← ^V(Successor(b))**



In the above figure, V\_train(b1) ← ^V(b3), where b3 is the successor of b1. Once the game is played, the training data is generated. For each training example, the V\_train(b) is computed.

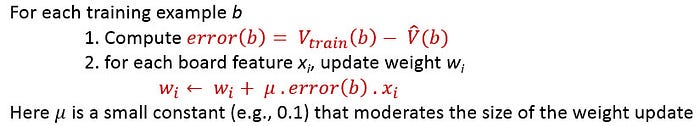
**Adjusting the weights**

Now its time to define the learning algorithm for choosing the weights and best fit the set of training examples. One common approach is to define the best hypothesis as that which minimizes the squared error E between the training values and the values predicted by the hypothesis ^V.



The learning algorithm should incrementally refine weights as more training examples become available and it needs to be robust to errors in training data  
Least Mean Square (LMS) training rule is the one training algorithm that will adjust weights a small amount in the direction that reduces the error.

The LMS algorithm is defined as follows:



**Final Design for Checkers Learning system**

1. The performance System — Takes a new board as input and outputs a trace of the game it played against itself.
2. The Critic — Takes the trace of a game as an input and outputs a set of training examples of the target function.
3. The Generalizer — Takes training examples as input and outputs a hypothesis that estimates the target function. Good generalization to new cases is crucial.
4. The Experiment Generator — Takes the current hypothesis (currently learned function) as input and outputs a new problem (an initial board state) for the performance system to explore.

**References —**

1. Machine Learning by Tom M. Mitchell
2. Learning from data by Yaser Abu-Mostafa

Reference

https://medium.datadriveninvestor.com/3-steps-introduction-to-machine-learning-and-design-of-a-learning-system-bd12b65aa50c