Machine Learning

Abhishek Ganesan

August 4, 2017

Abstract

This is a summary of all the major points touched in "Learning to see" by Welch Labs.

1 Introduction

What is Machine Learning? How can a machine possibly learn? Does it memorize already fed data and try replicating the same result for a somewhat similar problem?

How is it that the learning process for us humans seem so apparent, but the same seems like an impossible task for computers? Why can't machines do the same? How is it that something with such apparent simplicity be so complex practically?

Then came the famous computer scientist Alan Turing who came up with the proposal - anything that can be made into a mathematical model can be translated to a computer program and therefore be understood by the computer.

However, Artificial Intelligence (AI) was misunderstood. The AI problem was pretty much under-rated. We tried basing the identification of objects solely on a 8x8 grid of distinct numbers i.e, A particular sequence of numbers in the matrix was the equivalent of an object. This seemed to work to some extent.

Now came the next question, in what all fields can Machine Learning (ML) be of use to us.

1.1 Rules on rules on rules

More data would mean better performance? So we tried feeding more data (a.k.a Knowledge Engineering). This resulted in terrific improvement in the way data was processed. Things didn't go exactly as planned. The computer was able to process data already existing in its memory. But when new data was fed, the performance was disappointing, so disappointing we had to analyze a new approach. We thus reached a plateau. Do we keep feeding more rules and thereby data or try a new method of tackling this problem?

1.2 Artificial Intelligence

In 1980, Artificial Intelligence was re-introduced with the idea of commercializing it to make MONEY. It was looked at with a narrow perspective. Thereby, people were looking into only profitable sub-problems. Now people realized that Artificial Intelligence could be used for playing chess, proving mathematical problems etc., but thought that the process would be hard since it is hard for humans. Common thought among people was : "Problems hard for humans to solve would be hard for computers as well, and easier, easier". But in reality, it was the exact opposite. It was pretty obvious at this stage that Knowledge Engineering wasn't quite the way. This was the time for introspection. Time to question the most basic processes we do daily as humans. How do we think/learn? This alternate approach was pretty successful and is solely responsible for the growth of AI today.

1.3 Machine Learning

Dr. Arthur Samuel created a program that could learn to do a particular task. Not do the task, but learn to do the task. We were no longer interested in feeding rules to the computer, rather give examples (Note : examples could be rules also) and make the computer learn the rules thereby. Over time, this resulted in exponential growth. This is what we call today - Machine Learning.

Figure 1:

Knowledge Engineering was accurate but not correctly classified. There are two classifications :

- Recall
- Precision

Recall is the portion of data identified. Precision is the portion of all finger predictions that turn out to be correct.

Matching data to our existing data of fingers is what was used under the pretense of Machine learning, when in reality this was just memorization. It only gave astounding results to data already existing.

1.4 To learn is to generalize

Memorization is not the same as Learning. It is nowhere near learning.

The only way to assess the algorithm was to test it on unseen examples and measure performance. The more the algorithm was able to generalize, the more progress we made.

1.5 Rules

What are rules? How do we define what a rule is?

How do you predict whether a rule is correct or not? What if rules are redundant? What if one rule is a subset of an already existing rule? This will only increase the time complexity of the code.

The question to be asked was : "For any given problem, how many distinct rules do we need?"

1.6 Logical connectives

We started incorporating Logical Connectives (Boolean Algebra) to the already ongoing processes. We realized that any function can be broken down into sub-functions to an extent that any function could be represented using 3 basic gates -

- AND
- OR
- NOT

AND (or Intersection) is the mathematical equivalent of Multiplication.

OR (or Union) is the mathematical equivalent of Addition.

NOT (or Inverter) is the mathematical equivalent of Negation.

This helped us make a slight correction in the question already posed. Now, the question to be asked became : "For any given problem, how many distinct boolean functions do we need?" These functions could always be tested against our training data using Truth Tables.

1.7 More assumptions

Fundamentally, the only way to learn and make progress is by generalizing.

Could we also consider making educated guesses (or assumptions)? Because, the key to human learning is by making mistakes (in this case errors) and thereby learn from them. So instead of tailoring our algorithm to work against the training data, we should generalize by making a few assumptions.

1.8 Complexity of a rule

The only requirement for something to become a rule is its ability to generalize.

General rules are usually simple. They are also less in number to consider a whole lot more of data.

Emphasis was laid on robustness.

Another question posed at us was "How complex can a rule be?". How complex is complex?

1.9 Time complexity, a concern?

All these Boolean functions resulted in increasing the time complexity of our algorithm. Soon, time complexity became a limitation. Algorithm was becoming intractable and thereby causing concern.

1.10 Trees

This led to the introduction of Trees. Time complexity could be reduced drastically and this turned to be a major breakthrough. Time complexity was no longer a concern.

A greedy approach was adopted. This incorporated the idea of making educated guesses at every level hoping it will result in optimization at the end. By optimization, we mean to reduce the number of errors at every level so that the total number of errors reduces.

1.11 Heuristics

A heuristic approach could be the way? It is basically any approach to problem solving, learning that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals. Where finding an optimal solution is impossible(or impractical), heuristic methods can be used to speed up the process of finding a satisfactory solution.

1.12 Better heuristics

Considering the finger identification problem, impurity of all the pixels seemed equal. The impurity splitting rule is simple. It states Impurity before split SHOULD be greater than or equal to the Impurity after split.

1.13 Conclusion

- Knowledge Engineering approach wasn't useless entirely. It helped us understand what fingers look like.
- The entire concept of generalizing is ill-posed.
- Learning is basically the ability to balance between two strikingly different goals (performance and generalization)

To learn is to generalize!

1.14 References

- https://www.youtube.com/playlist?list=PLiaHhY2iBX9ihLasvE8BKnS2Xg8AhY6iV
- https://en.wikipedia.org/wiki/Machine_learning