Machine Learning,ML

Prof. Carl Gustaf Jansson
Prof. Henrik Boström
Prof. Fredrik Kilander
Department of Computer Science and Engineering
KTH Royal Institute of Technology, Sweden

Lecture 15
Genetic algorithm

Welcome to this fifth lecture the third week of the machine learning course. The topic for this lecture is genetic algorithms. So genetic algorithms is a specific an early representative for a class of computational models called evolutionary computing. As for all efforts in evolutionary computing it's inspired by theories and models from evolutionary biology, so genetic algorithms are commonly used to general global solutions to optimization /search problems. Genetic algorithm particularly useful for problem domains that have a very complex optimality landscape. The simplest form of genetic algorithms is based upon a representation of chromosomes, that's the data items in the form of such simple binary strings, with discrete functions for evaluating fitness of this chromosome fitness of the data, and syntactically defined breeding and mutation operators designed specifically for the binary strings. The idea of genetic algorithm could be engineered for much more complex representations on binary strings that's absolutely possible but that is not treated in this lecture.

So let's look at a moment, after this interdisciplinary sources of inspiration for this representation. So genetic elements are inspired by Darwinian evolutionary theory, an ideas of the survival of the fittest by natural selection. You can see below some pictures that illustrate the functionalities of what this gradual organic development of the natural species. So in Darwin's theory of evolution, populations changes over time and he was the first to propose a feasible mechanism for how this happen. So if we look at the core components now here so first of all what is the data set, the data set here is called the population and the data set consists of data items, so in this kind of framework the dietary terms are data items are called chromosomes where each chromosome in any way datum represents one potential solution to the problem at hand. And as I said earlier with what we discussed in this lecture it's a very simple model where each data items is a binary string and so this means that in this terminology used in genetic algorithm the gene is one position in such a string. So typically what we also do is a very basic operation is to rank each other chromosomes in a continuous fashion, in a regular fashion so this means every chromosome, every data item in every step of the process is evaluated by applying a fitness function, and of course the idea with all process is to produce in the end the fittest chromosome, one chromosome that has the best performance.

So what is the basic machinery of a generic genetic algorithm system then yeah the key issues are the following, so we already talked a little about that the key the basic representation so what happens is that we have this population of chromosome, then we have a computational cycle where in each step the fitness of all the whole population is evaluated and then after that a selection is made, subsets of the fittest from the population are chosen to directly be included in the next generation and the rest are discarded. Then and the selected ones are also allowed to mate, so actually pairs of the remaining ones, the ones who are fittest are allowed to mate to general children to restore the population site. This mating is typically carried out through a process called crossover which we will be I will describe a little later. After that one also allow a step where the new set of data items or chromosomes can be randomly mutated, when that takes place we have a new generation and then new cycle forms.

So one very important ingredients in the genetic algorithm machinery is the fitness function mentioned earlier, could also be called an object function or evaluation function. This function looks at every data item and evaluates how close that data item is to a given solution and because it has such a crucial role the success of an implementation of genetic algorithm us depend on this very clever choice or of the fitness function. It's very important, the few things are important it's very important that's clearly defined it's understandable. It's important that it generate intuitively reasonable results. It can be it, doesn't need to be simple it can be complex but it should be easy to understand and it should create intuitive results. It should also be such that it's efficient to implement because it will be evaluating many times so if you can consider you have a large population and you run this many times there is a computational issue here involved, and syntactically issue it is supposed to produce a quantitative measure real value that can discriminate the chromosomes as much as possible. So it was mentioned earlier that in the production of new chromosomes from parent chromosomes there are few key operations involved, so for mating between all chromosomes the crossover operation is used and then there is also mutation operation normally applied. So crossover them need some explanation, so single point crossover exactly essentially taking two strings, two parents choosing one point in the binary string and essentially swap the all the bits to the right of the power of that point. So as you see the examples the two parents left part up to the crossover point remains the same while the right strings are swapped giving then two children. And the alternative is to have multiple in crossover and which means that in the case of two-point cross of over two points are picked randomly and the section between the two points are swapped while the beginning and the end his remains the same and this can be generalized to key point but that's not it very important. Okay so this is crossover. And mutation operation is very simple it means simply to flip a bit at a random position. So the binary representation genetic programming scheme is a very simple straightforward and appealing, but there are a number of drawbacks with this and I want to mention a few issues. So it's crucial and non-trivial how the features of problems are mapped to binary strings. Already when we talked about neural networks we realized that it's not necessary a simple problem to take a set of features in some form and mapped them into the input layer of a neural network. In the genetic algorithm setting this becomes even more tricky because if you I hope understood that these chromosomes are changed and modified in a synthetic tactical manner which means that the crossover for example operations we break the binary string and there is various parts and it cannot happen of course that kind of cross over breaks apart bits that anyway belongs to the same feature. So the modelling of the feature set onto the chromosome binary strings is a non-trivial task. Also the choice of fitness function in is absolutely crucial. So the that is also an issue because it's a single component in the system and its behaviour it’s so important. Yeah and as I already touched depending on how the feature values are modelled onto the chromosomes one may have to restrict the crossover mutation operators, so they cannot produce non meaningful chromosomes from a problem domain point of view. Also here that a lot of parameters, hyper parameters that those exist for any system, I mention it for Neural networks but here there are many and these have to be adjusted typically, and those parameters are such as the size of reproduction subset, the mutation frequency, the various crossover policies etc. and then also a problem for this also for as for your networks that the amount of computation needed are our typical immense for for this kind of setup.

So I have included a more detailed example here and it's a very simple example and on this first slide you can see the setup so you have a number of binary digit constituting the chromosome of the size of eight we have four data item’s we have a function of Fitness function that counts the number of ones, and we have a crossover rate and which guides the crossover functionality we have a mutation rate and so on. So the hyper parameters are set and outline is done. So then in the next part of the example and the Fitness functions are evaluated and normalized and then the crossover operation is performed and then the mutation form which then creates actually a new population. And as is observed here it's not always that it always goes locally in the right direction it can be so that some of the fittest candidates disappear but then reoccur, actually when one very good property of genetic algorithm that is not sensitive for local optimization with which many other algorithms suffer from.

So for your convenience I included two other examples here, there are new aspects of these particularly importance it's just that you can see some of alternatives examples of the first, the first example is a solution to the Eight Queens problem, so as you can understand one issue here is to code the positions of the on the chess board of the eight Queens in a form so it can be mapped onto a chromosome. So in the second example you can see the problem is to maximize a numerical function, so this is just another kind of case.

So then the question is how does genetic algorithms relate to learning and of course there can be many possible connections here but, the one I want to mention at this point is called classifier systems. So essentially a classifier system is production system as it was described in one of the earlier lectures in the first lecture this week and essentially what one does in a classifier system is that the production rules of a production system are mapped onto the binary strings or a genetic algorithm. I knew already seen some sites ago you need to do this marking so for the 8 Queens problem you have to do it in some way, well maybe do another, so of course nothing prevents that you can take the symbolic rules of a rule based system or production system and map them onto the binary strings. So this means in a way that you can utilize the functionality of the unical going so that the population the data set is exactly a number of data items or chromosomes that are equivalent to an initial rule set that is supposed to solve a certain problem. So but then of course you need a separate model of this module in the system so where you in every generation apply this system the current generation to a specific problem and typically that module acts a lot like what was described in almost previous slides on rule based systems, so essentially there is a some kind of inference engines that tries out which satisfies, matches the problem at hand and of course you also mean then in the system some reinforcement feedback so that credit and blame can be given to individual contributing classifiers, and there is actually an argument algorithm for that compacted brigade every which is it's very similar to what in the neural network case is called back propagation.

So the feedback to the individual classifiers from the result of the application of the rule set for the basis for the fitness evaluation of the population, but for the rest you know you normal genetic algorithm machineries run which means that there will be a new generation rules every time the new unit will fight again to the problem there will be feedback generated that will be created a blame to the to the individual classifiers and so on and so on. So this is one approach to combining a learning scenario of a rule-based system to the genetic algorithm setup. So in this line it is just a repetition of what I said a little earlier, so essentially a classifier system is a combination of a generic algorithm running and a kind of rule-based reinforcement based system so essentially have a rule based the rule base you see in the picture the constitutes the population genetic algorithm. But in every step of the process this rule based is applied to a certain problem task and the result of that problem solving is then fed back in into the into the root system again so that credit and blame are given to the individual rules and then the credit and blame the distribution of the rules are used as the basis for the fitness function and then the genetic algorithm creates a new direction and so on. So this was the end of the fifth lecture this week, so the six lecture number six will be on the topic of logic programming. thank you goodbye