P3: We did a thing

Sara Pernille Jensen & Håkon Olav Torvik

FYS-STK4155 – Applied Data Analysis and Machine Learning
Autumn 2021
Department of Physics
University of Oslo

December 3, 2021

ABSTRACT: The very concrete abstract.

Contents

1 Introduction

The theoretical basis for machine learning and artificial intellegence was developed many decades ago, with the term *machine learning* being coined in 1959. But it is only in the last decade or so where deep learning have seen widespread use, making complex data analysis possible. Advances in computing power is one of the main reasons for this, but also refining of the statistical methods have been important in yielding good results. Today, neural networks form an integral part of many algorithms, being able to solve new problems, or old problems more efficiently. In [2] and [3], feed forward neural networks were used to both fit continous data, as well as classification.

Partial differential equations with many independent variables are notoriously hard to solve analytically. The Navier-Stokes equations, describing the motion of viscous fluids, is a famous equation with no known exact solution. There exist numerical schemes to solve it in certain cases. An alternative could be to use neural networks. This paper will show that machine learning can solve PDEs. Specifically, the equation studied is the one-dimensional diffusion equation. First, a feed forward neural network is used. Secondly, a genetic algorithm building on the theory of natural selection is applied on the same problem. It will also be solved analytically and with an explicit scheme to compare accuracy and efficiency.

Further, Yi et. al. showed that the eigenvalues of a symmetrix matrix can be described by a differential equation, solvable by a neural network [6]. These findings will be replaced here, displaying an application of neural network-differential equation solvers.

2 Theory

The temperature gradient $u(x,t)$ in a rod of length $L = 1$ can be described by the one-dimensional diffusion equation, a partial differential equation with two independent variables, on the form

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}. \qquad (2.1)$$

,

where x is the length along the rod, and t is time. The boundary and intial conditions are

$$\begin{aligned} u(0,t) &= 0 \\ u(L,t) &= 0 \qquad\qquad (2.2) \\ u(x,0) &= sin(\pi x) \end{aligned}$$

It is easily verifiable that the specific analytic solution is

$$u(x,t) = e^{-\pi^2 t}\sin(\pi x)$$

2.1 Neural Network

A feed forward neural network consist of L layers, each with a set number of nodes. The layers are connected by weights and biases P, such that each node takes the value of a weighted sum of the nodes in the previous layer. An activation function is used to regularize the value each node can take. Some input is placed in the input layer, and from where it is fed forward through the network, until it reaches the output layer. This is a prediction of the input data, which can be the funcion value of a continous function at a certain point, or wether a tumor is malginent or not. Under supervised learning, this prediction is compared to a true value to find the error. The backprogation algorithm then calculates the gradients of the

weights and biases which will minimize the error, and update them appropriately. Over several iterations, the network can become very accurate, and successfully predict results from new data. A full description of the architecture and inner workings of a feed forward neural network, along with discussion of strenghts and weaknesses associated with such a method, can be found in [2] and [3].

In order to determine the error, a cost function must be defined. In the case of regression problems, this could simply be the mean of the error in all points, squared. For classification, cross entropy is used. When solving a differential equation, a natural choice is the difference in the left and right hand side of the equation, squared. Trenger å se lecture notene til morten for å skrive rett notasjon her.

2.2 Genetic Algorithms

Genetic algorithms form a family of machine learning algorithms inspired by the process of natural selection. The use of evolutionary systems to develop computational methods to solve optimisation problems stems back to the 1950s and 60s. Genetic algorithms were originally developed by John Holland and his and colleagues at the University of Michigan in the 60s and 70s, and his main theoretical framework as presented in his 1975 book *Adaptation in Natural and Artificial Systems* [1] is still in use today.

The main idea behind genetic algorithms is to take inspiration from the unsupervised Darwinian evolution of populations of living organisms over generations and create an analogous machine learning algorithm. There are three main principles of Darwinian evolution which are usually considered the necessary and sufficient conditions for natural selection to occur. These are a) phenotypic variation, b) differential fitness, and c) heritability. Phenotypic variation is the population level property of there being variation in the phenotypic traits of the individuals in that population, and not just genotypic variation. Differential fitness is the property that survival and reproduction rates vary between the individuals in the population, and heritability is the property that some traits are passed onto offspring from their parents.

Over the generations, the individuals in the population gradually become better adapted to the environment in which they live as a result of cumulative selection. This requires that not only the final change is favourable, but the adaptive shift at each step must have provided an improvement in order for it to have been selected for. This idea is well represented by the idea of "fitness landscapes", introduced by Sewall Wright in the 1930's [5]. These landscapes, also known as adaptive landscapes or evolutionary landscapes, provide a visualisation of the link between the genotypes (the complete set of genes of an individual) or phenotypes (the observable traits of an individual) and the individual's adaptive fitness or success. The adaptive success of the individual is represented by the height of its position in the landscape, and individuals with similar genotypes are expected to be located nearby each other in the landscape. These landscapes can be multidimensional, although the exact parameters along the other axes will be complex functions of the environment, and are rarely studied in detail. Instead, such landscapes provide a useful metaphor and tool for visualising evolutionary optimisation, where the population as a whole is expected to move towards higher points in the landscape over time. Furthermore, it makes it easy to understand how populations can end up at local minima with suboptimal traits which evo-

lution struggles to "improve", since moving to the true optimum would require an initial descent in the landscape, which cannot happen through cumulative selection. Random mutations can in some cases help to prevent this, but not always.

Applying this to machine learning, it is common to initialise a "population" of "chromosomes", which form a set of candidate solutions. Each chromosome is encoded as a list of genes following a set of encoding and decoding rules determined by the algorithms. Each gene then encodes an element of the candidate solution. For each generation, the "fitness" of each chromosome is evaluated based on some metric which is summarised in the cost function. The fitness of the individuals then determines their chance of reproduction. A new generation of chromosomes is then generated through reproduction (crossover) and mutation, where the fittest chromosomes from the past generation are most likely to have their genes passed on. A range of different algorithms can be used for these two steps. Over the generations, it is hoped that the fitness of the candidate solutions will increase until a sufficiently good model is reached. However, the concept of fitness landscapes apply equally well to such machine learning algorithms as to the evolutionary process. There will be always be a significant risk of getting stuck at local maxima in the landscape, where the evolutionary process stagnates at a sub-optimal solution. Getting stuck in such local extrema is an ever-present problem in all of machine learning. Genetic algorithms thus stands out in that the mutation operator helps at getting out of these stagnation points, increasing the chances of reaching the global maxima in the fitness landscape.

2.2.1 Differential Equations and Grammatical Evolution

Genetic algorithms have been found to be useful in various optimisation problems, as suggested by the importance of the notion of the fitness landscape in the framework. As discussed above, it is possible to reformulate differential equations as optimisation problems. The use of genetic algorithms in solving such problems is a fairly recent development. It can be done in various ways, but in the present paper the focus shall be on a method based on *grammatical evolution*, first presented by Tsoulos and Lagaris in their [4]. Here, each chromosome is an analytic expression encoded by the genes through an encoding scheme. This scheme is given by the grammar of the algorithm. If the solution can be expressed in closed-form, there is thus hope of retrieving the true solution to the problem. If not, an approximate solution can be found. The algorithm and grammar here used is heavily inspired by the one used by Tsoulos and Lagaris in their [4].

3 Methods

3.1 Genetic Algorithm

The outline of the algorithm is as follows. A population-class is defined containing a list of C chromosomes with randomly generated sequences of G genes. For each generation, the genetic sequence of each chromosome is decoded and its corresponding analytic expression is generated. The fitness value of each chromosome is then calculated, and the list of chromosomes ordered according to their fitness. Chromosomes whose fitness is `nan` or `inf` are removed from the list before crossover. The E fittest individuals are then passed onto the next generation unaltered, whereas the remaining new individuals are generated according to some

crossover scheme, before mutations are applied to some of these. This process is repeated until the maximum number of generations is reached, or until the fitness criteria is met by the fittest chromosome.

3.1.1 Grammar

Each chromosome contains 50 genes, where each gene is an integer value in the range [0, 255]. A chromosome is interpreted by reading the genes sequentially and constructing an analytic expression using the grammar as given in Table 1. The meaning of a given gene thus depends on the type expected when it is interpreted. To decode it, the modulus of the gene with the number of options in that type is taken. E.g., if an expression `<expr>` is expected, the gene 17 will be interpreted as 17 mod 6 = 5, giving "t". Note that the decoding algorithms always interprets the first gene as an expression `<expr>`. Furthermore, to prevent the candidate solutions from being too simple, the first gene in the chromosomes was always set to 0 or 2.

The analytic solution to the problem studied is encoded as: [0, 2, 2, 0, 3, 0, 1, 0, 3, 10, 2, 0, 3, 10, 2, 5, 2, 2, 0, 0, 3, 10, 2, 3].

3.1.2 Fitness evaluation

The fitness of an individual is a weighed sum of its deviance from the differential equation and from the boundary conditions. Only the equations for partial differential equations will be included here, but these are easily simplified to other types of differential equations.

The ranges and number of collocation points, N, for each variable x and t must be given. The same number of points were used for both variables. The fitness is calculated by adding the squared residuals of the collocation points for the differential equations and the squared error at the boundaries. Since

Table 1. Encoding scheme defining the grammar of the algorithm, showing how the genes are sequentially read to generate an analytic expression.

Type	Options	Index
`<expr>`	`(<expr><op><expr>)`	[0]
	`(<expr>)`	[1]
	`<func>(<expr>)`	[2]
	`<digit>`	[3]
	`<x>`	[4]
	`<t>`	[5]
`<op>`	+	[0]
	−	[1]
	*	[2]
	/	[3]
`<func>`	`sin(<expr>)`	[0]
	`cos(<expr>)`	[1]
	`exp(<expr>)`	[2]
	`x**(<expr>)`	[3]
	`t**(<expr>)`	[4]
`<digit>`	0	[0]
	1	[1]
	2	[2]
	3	[3]
	4	[4]
	5	[5]
	6	[6]
	7	[7]
	8	[8]
	9	[9]
	π	[10]

the squared residual will always be a positive number, and the optimal model will have a residual of 0, to keep with the metaphor from the fitness landscape of the fittest individual having the highest fitness score, the negative is used. Thus, the true solution has a fitness of 0. Define each candidate solution is labelled $M_i(x, t)$ for $x \in [x_0, x_1]$ and $t \in [t_0, t_1]$, both with N points.

For a function $u(x, t)$, the differential equation can be expressed as is expressed in the form:

$$f\left(x, t, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial t^2}\right) = 0$$

The contribution to the fitness from the partial differential equation is then given by:

$$D(M_i) = -\frac{1}{N^2} \sum_{k,l=1}^{N} f^2$$

In the case of the diffusion equation:

$$D(M_i) =$$
$$-\frac{1}{N^2} \sum_{k,l=1}^{N} \left(\frac{\partial^2 M_i(x_k, t_l)}{\partial x^2} - \frac{\partial M_i(x_k, t_l)}{\partial t}\right)^2$$

The factor of $1/N^2$ is not necessary, but it normalises the fitness to make solutions obtained with different number of collocation points comparable.

The contribution to the fitness from the boundary conditions is given by:

$$B_1(M_i) = -\frac{1}{N} \sum_{l=1}^{N} (M_i(x_0, t_l) - u(x_0, t_l))^2$$

$$B_2(M_i) = -\frac{1}{N} \sum_{l=1}^{N} (M_i(x_1, t_l) - u(x_1, t_l))^2$$

$$B_3(M_i) = -\frac{1}{N} \sum_{k=1}^{N} (M_i(x_k, t_0) - u(x_k, t_0))^2$$

$$B_4(M_i) = -\frac{1}{N} \sum_{k=1}^{N} (M_i(x_k, t_1) - u(x_k, t_1))^2$$

In the given problem:

$$B_1(M_i) = -\frac{1}{N} \sum_{l=1}^{N} (M_i(0, t_l))^2$$

$$B_2(M_i) = -\frac{1}{N} \sum_{l=1}^{N} (M_i(L, t_l))^2$$

$$B_3(M_i) = -\frac{1}{N} \sum_{k=1}^{N} (M_i(x_k, 0) - sin(\pi x_k))^2$$

The total fitness is then:

$$F(M_i) = D(M_i) + \lambda(B_1(M_i) + B_2(M_i) + B_3(M_i) + B_4(M_i))$$

Where the parameter λ is a penalising parameter which can be changed to tune the penalising contribution from the boundary conditions compared to the differential equation. This can be useful if one factor contributed much more to the fitness than the other.

3.1.3 Reproduction

There exists a range of different algorithms for creating the next generation of chromosomes from the current generation. The main elements of the algorithm for reproduction are the selection, crossover and mutation schemes used.

First, elitism was used before any of the other reproduction operations. This involves a predetermined number E of the fittest individuals to be passed onto the next generation unaltered (without mutations), whilst still being used for reproduction. This ensures that the highest fitness value never decreases between generations.

The selection scheme determines what chromosomes from the current are chosen as par-

ents. Two different methods were here investigated; probabilistic selection and tournament selection. Assume the current generation contains C individuals and NC new chromosomes are to be created. For probabilistic selection, a list containing $2NC$ integers distributed according to half a normal distribution centred at 0 with standard deviation $0.2C$ is generated. These number are then iterated over in pairs, and are used as indices to choose which two individuals from the sorted list of chromosomes are to be chosen as parents. This ensures that the fittest individuals have a much higher chance of reproduction, but genotypic and phenotypic variance is still ensured by allowing the less fit individuals to reproduce some of the time. Tournament selection involves picking K random integers in the range [0, C] and pick out the lowest two. These are then used as the indices to pick out the two parents from the sorted list of chromosomes. Note that both methods allows the same chromosome to be chosen as both parents, in which case the child will always be a perfect copy of the parent(s) before mutation. How the new chromosomes are generated from the two parents is then determined by the crossover scheme.

The crossover scheme determines how new chromosomes are created from the past generation. Three different methods were investigated here. First, one can create a new genetic sequence which is a perfect mix of the genes of the two parents. $0.5G$ different integers are picked from the range [0, G], and these are used as the indices of the genes used from one parent, and the remaining genes are taken from the other parent at the loci not picked by the indices. Note that the loci of the genes in the sequence are always preserved. This is the method most analogous to how reproduction takes place

in nature. Alternatively, a random index in the range [0, G] is chosen, and this is used as the crossover point, such that the genes before that point is taken from one parent and the genes after that point from the other parent. The two halves are then combined to create a new genetic sequence.

After the crossover operation, a mutation operation is applied. Two rates are important in choosing how this is done. Firstly, one must decide what percentage of the new chromosomes are to have their genes mutated, and secondly how many of the genes are to be changed. The mutation operation itself works by randomly picking a given number of indices in the range [1, G] (the first gene was always constrained to be 0 or 2) to determine the loci of the genes to be changed. The genes at the chosen loci are then replaced by random integers in the range [0, 255].

4 Results

5 Discussion

6 Conclusion

References

[1] John H Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* Complex adaptive systems. MIT Press, Cambridge, Mass, 1992.

[2] Sara Pernille Jensen and Daniel Johan Aarstein. Regression and classification, using stochastic gradient descent, feed forward neural networks and logistic regression. *University of Oslo*, October 2021.

[3] Håkon Olav Torvik, Vetle Vikenes, and Sigurd Sørlie Rustad. Machine learning: Using regression and neural networks to fit continuous functions and classify data. *University of Oslo*, October 2021.

[4] I.G. Tsoulos and I.E. Lagaris. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 7:33–54, 2006.

[5] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the XI International Congress of Genetics*, 8:209–222, 1932.

[6] Zhang Yi, Yan Fu, and Hua Jin Tang. Neural networks based approach for computing eigenvectors and eigenvalues of symmetric matrix. *Computers & Mathematics with Applications*, 47(8):1155–1164, 2004.