COMP305 First Semester Examination 2020/21

Biocomputation

ID: 201277571

01/02/2020

1 History and Concepts

1(a)

Genetic Algorithms (GA) and Artificial Neural Networks (ANN) are used to solve a variety of computer science problems. Initially based on biological phenomena, such as evolution and neurons, the techniques are now applied to problems spanning computer science, engineering, economics and more. From predicition [1], to optimisation [2], these biology-inspired theories are represented algorithmically to solve ground breaking computer science problems. These problems can be biology related (e.g predicting cancer survival rates [1]), or completely removed from biology (e.g spacecraft antenna development [2]). The prolific use of these theories within Computer Science and in situations completely removed from their original biological context, warrants the inclusion of these theories as part of Computer Science.

1(b)

ANN can be used to solve problems in a range of disciplines including optimisation, prediction, classification, and pattern recognition. The McCulloch-Pitts neuron was the first proposed computational model of a neuron and first piece of work to be recognised as AI [3]. Many important developments in AI were achieved using ANN as a foundation.

An example of the use of ANN to classify individual handwritten digits can be shown on the MNIST dataset [4]. A sample of the 28*28 pixel, greyscale dataset can be seen in figure 1

Figure 1: A small sample of the MNIST dataset

As the images are in greyscale, each pixel in the image can be represented as a numeric greyscale value in an array. These arrays can then form the input to a multi-layered perceptron. Depending on the specific architecture used, the model can "learn" what each digit looks like and accurately classify digits it has not seen before. [1]

[1] This is an extreme oversimplification of the complexity of multilayer perceptrons, even in a seemingly simple application such as this.

Character recognition such as this has a wide variety of applications, from automatically reading numbers on forms to verifying signature authenticity.

1(c)

Genetic algorithms are used to solve optimisation problems spanning a range of disciplines. Inspired by the process of natural selection, they form a subclass of the wider class of evolutionary algorithms. A population of "chromosomes" represents a population of candidate solutions to a given optimisation problem. A selection of genetic operators (including mutation, crossover and inversion) are applied to the chromomes and mimic the process that occurs during natural selection.

An example that shows the connection between Genetic Algorithms and their origin in natural selection, can be seen in the following paper [5] and figure 2 below.

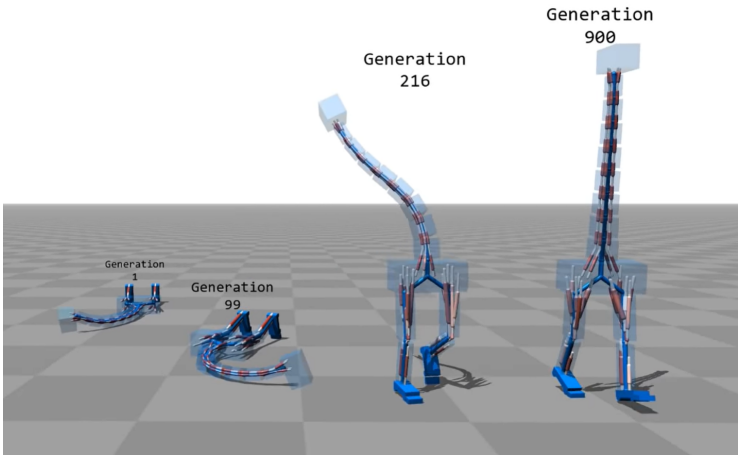

Figure 2: Video still, taken from the video at [6]

In this paper, various optimisation algorithms are used. Most notably in this instance, is the use of the Covariance Matrix Adaptation [7], an evolutionary algorithm. Figure 2 shows how, over the course of many generations, the chromosomes [2] in a population are optimised towards an objective.

Other applications of genetic algorithms include spacecraft antenna construction [2], travelling salesman problem [8] and boundless others. Despite the wide array of problem disciplines, there are some constraints as to which problems are suitable. For example, problems must have non-differentiable fitness functions and be able to be coded into a string of characters so the genetic operators can be applied.

1(d)

i. The OCR system is solving a pattern recognition problem. If the target outputs are listed as classes, it can be seen as a classification problem.

[2] In this case, the choromosomes in question are the values input into the simulated muscles

At it's simplest form, this can be seen in the many models trained on the MNIST dataset, mentioned in section 1(b), where inputs are classed as a digit between 0 and 9.

ii. There are two main types of OCR algorithm which both take different inputs.

Matrix matching aims to classify the shapes it identifies in the document as individual characters. It does this using an array value of the shapes and classifying this as a character, based on it's training data - much like MNIST. Any digit from figure 1 could be an example input, or any individual character on this page. Individual characters such as these are the ideal input however the system could also pick up any other shapes in the document.[3]

Feature Extraction is another type of OCR algorithm that simplifies the document to a series of features; mainly lines, curves, and intersections. This has many benefits including: the dimensionality of the input is reduced, making the process more efficient; the software can more accurately a variety of font and handwriting styles. Various AI methods can be used to compare the features to those the model is trained on.

iii. There is an intended character/ string for each character/ string being read by the OCR into the system. The training and test data is all labeled. Therefore, this is supervised learning.

However, there have been recent developments in the area of unsupervised text recognition [9]. This paper from the Visual Geometry Group at the University of Oxford proposes a method of recognising text in images without labelled traiing data. This combats the time and energy intensive process of manually labelling training data.

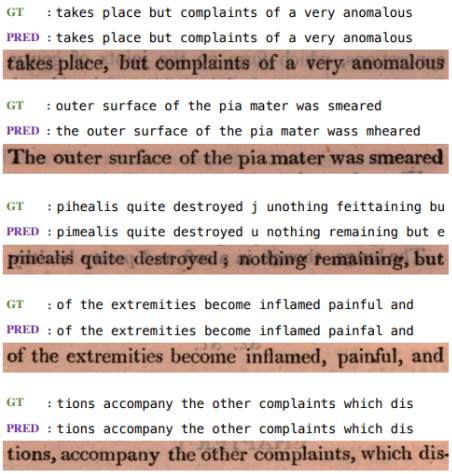

Figure 3: Example results of the model proposed in [9]

[3]In order to limit the possibility of this occuring, the document will likely undergo pre-processing. Noise can be removed, pages can be aligned and the contrast can be increased to make classification easier.

As shown in figure 3, there were varying levels of success. Despite this, this is an incredible advancment in text recognition and shows that unsupervised learning may be the future of text recognition. Or a worthy competitor at the very least.

iv. Ignoring the possibility of unsupervised learning, mentioned previously, the network would need to be trained using labelled data. This would likely be in the form of images of characters (handwritten or typed) with their correct label. The NIST dataset (a handwritten character dataset from which MNIST is found) is a good example. Data could also include previously translated and validated documents, any image of text which has an assosciated translation/ label could be used to train the network. The ability for the network to classify characters depends heavily on the training data used. For exammple, a network trained on typed fonts would likely struggle when presented with handwritten characters.

2 The McCulloch-Pitts neuron

2(a)

See figure 4, below.

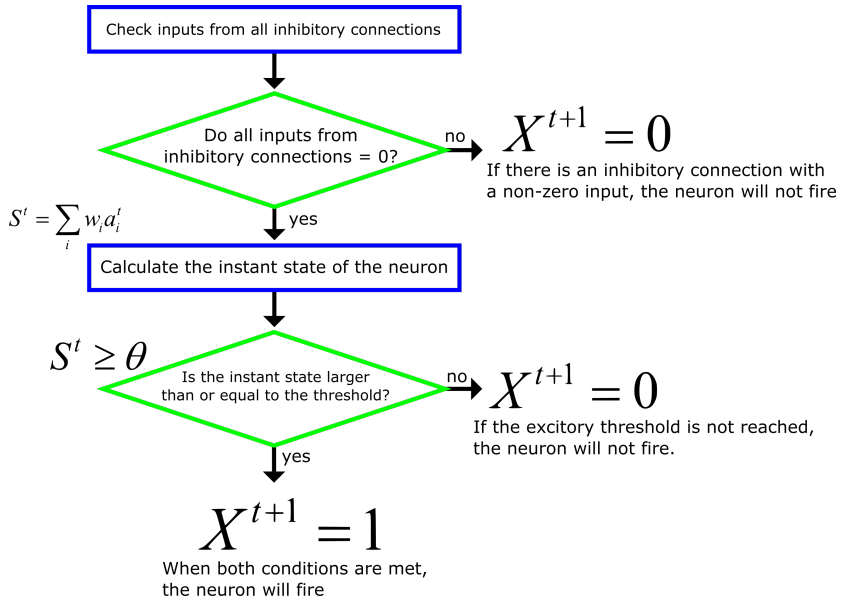

Figure 4: A flow chart diagram of an MP neuron where all weights of connections and the neuron threshold are set up in advance.

2(b)

Represented in boolean algebra as $a_1 \vee a_2$, and table form below, the logical OR gate can be represented as an MP-neuron as in figure 5.

a_1	a_2	"OR"
1	1	1
1	0	1
0	1	1
0	0	0

This table shows all possible outputs of the OR gate depending on all possible input values of a_1 and a_2. If either of a_1 and a_2 is "true" (has an input of 1), then the neuron will fire ($X = 1$).

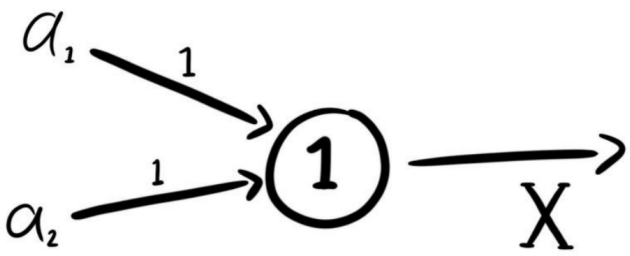

Figure 5: A diagram of an MP-neuron realisation of an "OR" logical gate

For this OR gate, the excitation threshold (θ) must be $0 < \theta \leq 1$. The value of 1 is chosen in figure 5, but any value within the given range would be sufficient. This inequlity can be proven to be correct by analysing the output for each possible value of a_i (assuming the weights remain at 1).

The following values of a_i should fire the neuron ($X = 1$) when the excitation threshold is 1 ($\theta = 1$) as at least one of a_1 and a_2 is always true (equal to 1).

$a_1 = 1, a_2 = 1 \Rightarrow a_1 + a_2 = 1 + 1 = 2, 2 \geq \theta \Rightarrow X = 1$

$a_1 = 1, a_2 = 0 \Rightarrow a_1 + a_2 = 1 + 0 = 1, 1 \geq \theta \Rightarrow X = 1$

$a_1 = 0, a_2 = 1 \Rightarrow a_1 + a_2 = 0 + 1 = 1, 1 \geq \theta \Rightarrow X = 1$

Conversley, if neither of a_1 and a_2 are true then excitation threshold should not be reached and the neuron will not fire.

$a_1 = 0, a_2 = 0 \Rightarrow a_1 + a_2 = 0 + 0 = 0, 0 < \theta \Rightarrow X = 0$

The above functions show that the inequality of the excitation threshold must be $0 < \theta \leq 1$.

2(c)

Represented in boolean algebra as $\neg a_1$, and table form below, the logical OR gate can be represented as an MP-neuron as in figure 6.

a_1	"NOT"
1	0
0	1

This table shows all possible outputs of the NOT gate depending on all possible input values of a_1.

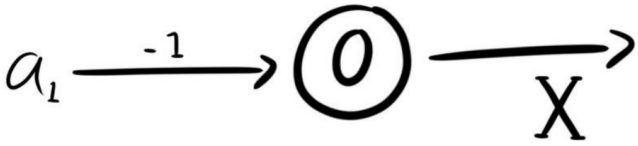

Figure 6: A diagram of an MP-neuron realisation of a "NOT" logical gate

For this NOT gate, the excitation threshold (θ), must be $\theta = 0$. This threshold value can be proven to be correct by analysing the output for each possible value of a_1.

If a_1 is true ($a_1 = 1$), it causes in input from an inhibitory connection, therefore the neuron will not fire. If a_1 is false ($a_1 = 0$), then the threshold is met as $0 \geq \theta$. The neuron will fire.

2(d)

The first neuron is an OR gate. Since both a_1 and a_2 are 0, it will not as the threshold of 1 is not met. This can also be seen in table format in question 2(b). This is followed by a NOT gate. Since the first neuron has not fired (providing an input of 0), this neuron will fire as the threshold of 0 is met. This logic can be seen in the NOT gate in question 2(c). As this is the final gate, and it has fired, the output is $X = 1$ (The neuron fires).

These answers can also be found by following the flow chart shown in part 2(a), as seen in figure 7

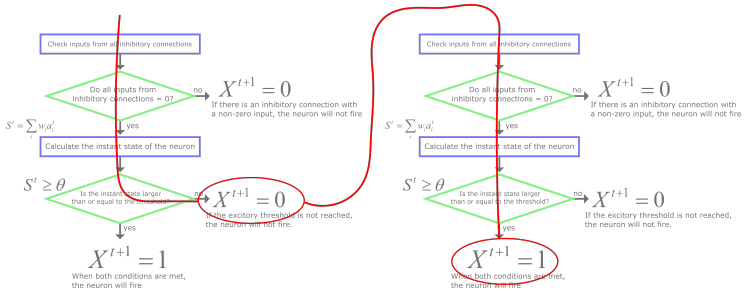

Figure 7: The route taken through the flow chat for the network in question 2(d)

2(e)

Figure 8 shows the MP-neuron representation of a memory cell. A memory cell, such as this, can be formed by closing a neuron in a feedback loop. To initialise this cell, an excitory input (a_1 in this case) is fired. This causes the value of 1 to cycle the loop (repeatedly fire the neuron), essentially storing it in the cell's memory. Even if a_1 were to fire again, the loop would still be constant at 1. That is, until the inhibitory connection fires. a_2 firing would cause the neuron to not fire, therefore breaking the cycle.

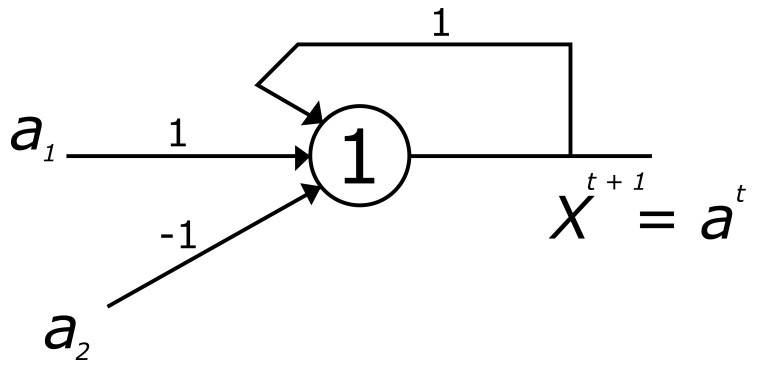

Figure 8: An MP-neuron representation of a memory cell

3 Learning rules of the Artificial Neural Networks. Hebb's Rule.

3(a)

Learning rules define how the weights of connections between neurons can be changed. If used during unsupervised learning, the aim is to reflect the network experience. For supervised, the aim of changing the weights is to get a desirable output. If unsupervised, the aim is to reflect the network experience.

There are a variety of different learning rules that can be used on ANN. These include: the Hebbian rule which identifies how the weights of a network can be modified, the correlation rule, the Delta rule and more.

3(b)

In his 1949 book, 'The Organization of Behaviour' [10], Hebb introduced the Hebbian Theory. He proposed that a particular type of use-dependent modification of the connection strength of synapses might underlie learning in the nervous system.

His formula can be simplified to the following method: The weight of connection at every next instance is increased such that:

$$w_{ji}^{k+1} = w_{ji}^{k} + \Delta w_{ji}^{k},$$

where

$$\Delta w_{ji}^{k} = C a_i^k X_j^k.$$

In these equations,
w_{ji}^{k} is the weight of the connection at an intant k.
w_{ji}^{k+1} is the weight of the connection at the next instant $(k+1)$.
Δw_{ji}^{k} is the increment that the connection weight is increased by.
C is the positive coefficient that determines learning rate.
a_i^k is the input value from the presynaptic neuron at an instant k.
X_j^k is the output of the postsynaptic neuron at the same instant k.

The weight of a connection changes iff the preceding input via the connection and the resulting output are simultaneously both $\neq 0$.

3(c)

The correlation nature of a Hebb synapse is shown in the equation

$$w_{ji}^{k+1} = w_{ji}^{k} + \Delta w_{ji}^{k},$$

For this reason is can be referred to as the activity product rule.

3(d)

The Hebbian rule represents unsupervised learning of an artificial network. There is no target output, and the weight of a connection changes according to the neuron's 'experience'. As seen in section 3(a), this describes a learning rule for unsupervised learning.

3(e)

time step	a_1	a_2	a_3	w_1^t	w_2^t	w_3^t	X	Δw_1	Δw_2	Δw_3	w_1^{t+1}	w_2^{t+1}	w_3^{t+1}
1	1	0	0	-1	2	-1	0	0	0	0	-1	2	-1
2	0	1	0	-1	2	-1	1	0	0.25	0	-1	2.25	-1
3	1	1	1	-1	2.25	-1	0	0	0	0	-1	2.25	-1

i. At $t = 1$, an inhibitory connection is fired (a_1 with $w_1^t = -1$) so the value for X was 0. If $X = 0$, then all values of Δw_i will be 0 because they are formed from a multiple of X. As a result of this, all the weights will be the same in the next time step.

ii. At $t = 2$, no inhibitory connections are fired and the sum of the fired inputs is greater than the threshold ($\theta \geq 1$). The neuron will therefore fire so $X = 1$. This can be used to compute $\Delta w_2 = 0.25$ but the remaining two Δw_i are still 0 as they are multiples of their inputs which are also 0. The Δw_2 value can then be added to the old weight to form the weight for the next timestamp.

iii. $t = 3$ is similair to $t = 1$ in that there is an inhibitory connection so $X = 0$. For the reasons explained in 3(e)i), the weight values will all carry through to the next timestamp. There will be no increase/ decrease in the weight values as all $\Delta w_i = 0$.

4 N/A

5 N/A

6 Genetic Algorithms.

6(a)

The famous book, 'On the Origin of Species' [11], written by Darwin in 1859 is considered the foundation of evolutionary biology. He introduced the idea that

species can adapt to an environment using natural selection and survival of the fittest.

Each individual member of the population is tested by environmental and social challenges. The concept of all members being individually tested in parallel is obviously an exciting concept to computer scientists as it promises a faster optimisation of a population than a sequential method.

The population must individually adapt to the environment, resulting in an overall optimised population. This can easily be seen mirrored in genetic algorithms as a population of candidate solutions faces individual adaption (by genetic operators) to optimise the population as a whole.

Due to natural selection and survival of the fittest, only the candidates that are best fit to the environment survive and reproduce to pass on their genes to the next generation. This selection process causes optimisation on both an individual and population-wide level.

The success of paralellism, adaption and optimisation in a natural environment is strong encouragement that these concepts could be useful in a computer science setting. In the 1950s - 60s, the first experiments into using evelotionary algorithms to solve computer science problems were being conducted. Genetic algorithms themselves, were not realised until Holland and his colleagues work on the subject in the 1960s. He first presented genetic algorithms in his 1975 book: Adaptation in Natural and Artificial Systems [12].

6(b)

1. Randomly generate a population of chromosomes (n bit strings)
2. Evaluate the fitness of each chromosome i.e the number of 1s in the string.
3. Repeat until next generation of n individual chromosomes are produced:

   a) Select a pair of parent chromones. The chromosomes with higher fitnesses should be selected most often.

   b) Apply crossover with probability

   c) Apply mutation with probability of occurrence
4. Apply generational replacement.
5. If termination condition is not met, return to 2.

Step 1 attempts to emulate a natural population as chromosomes while step 2 is evaluating the 'fitness' of each indervidual (survival of the fittest). Step 3 then shows the process of surviving and reproducing to produce a second generation, ideally more optimised than the previous. Step 4 speaks for itself and the cycle is repeated (as is the process of natural selection) until the optimisation goal is achieved.

6(c)

Schemas, often called the building blocks of the chromosome, are similarity templates that describe a subset of strings with similarities at different positions.

They utilize the characters 0, 1 and * where a 1 or 0 means that the corresponding but must be the same. A * (or #) is a 'do not care' symbol so in this case, either a 0 or 1 could be a match. For example, 11*1 could match 1101 or 1111.

To match a chromosome of length N, the scheme must be the same length and each symbol must either be the same as the symbol in the chromosome at the same location or a do not care symbol (*). Therefore, for a chromosome of length N there are 2^N schemas. For example, the chromosome 100 has $2^3 = 8$ schemas.

The order of a schema is the number of symbols it contains that are not 'do not care' symbols. For example, the schema 110**0 has an order of 4.

The defining length is the distance between the outermost non-'do not care' symbols so 2 for **0110* and 0 for *11*. You can compare defining lengths to each other. For example, the defining length of *11* is less than that of **0110*.

6(d)

schema	order	def. length
IQ	2	0
*Q	1	0
I*	1	0
**	0	0

6(e)

i. The average fitness of the schema 011* can be calculated using all possible strings for that schema:

$0110 \Rightarrow f(0110) = 2$

$0111 \Rightarrow f(0111) = 3$

The average can then be found:

$(2 + 3)/2 = 2.5$

So the average fitness of the schema $f(011^*) = 2.5$

ii. Using the same method as above, fitness of all possible bit strings are listed below:

$0101 \Rightarrow f(0101) = 2$

$0100 \Rightarrow f(0100) = 1$

$0001 \Rightarrow f(0001) = 1$

$0000 \Rightarrow f(0000) = 0$

$(2 + 1 + 1 + 0)/4 = 1$

As such, the average fitness of the schema $f(0^*0^*) = 1$

References

[1] R. Xu, X. Cai, and D. C. Wunsch, "Gene expression data for dlbcl cancer survival prediction with a combination of machine learning technologies," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. IEEE, 2006, pp. 894–897.

[2] J. D. Lohn, G. S. Hornby, and D. S. Linden, *An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission*. Boston, MA: Springer US, 2005, pp. 301–315. [Online]. Available: https://doi.org/10.1007/0-387-23254-0_18

[3] *Artificial Intelligence: A Modern Approach. Third Edition*. England: Pearson, 2016.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.

[6] T. Geijtenbeek, *Flexible Muscle-Based Locomotion for Bipedal Creatures*, 2013. [Online]. Available: https://www.youtube.com/watch?v=pgaEE27nsQw&feature=emb_title&ab_channel=ThomasGeijtenbeek

[7] N. Hansen, *The CMA Evolution Strategy: A Comparing Review*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. [Online]. Available: https://doi.org/10.1007/3-540-32494-1_4

[8] H. Braun, "On solving travelling salesman problems by genetic algorithms," in *Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 129–133.

[9] A. Gupta, A. Vedaldi, and A. Zisserman, "Learning to read by spelling: Towards unsupervised text recognition," in *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, 2018, pp. 1–10.

[10] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[11] C. Darwin, *On the Origin of Species by Means of Natural Selection.* London: Murray, 1859, or the Preservation of Favored Races in the Struggle for Life.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975, second edition, 1992.