Universidade do Minho - *Campus* de Gualtar

Machine Learning 101
CBR, GA and SVM

Group 5
Floris Bogaert - E7897
Miguel Calafate - PG35405
Pedro Alves - PG36091
Rita Canavarro - A74484

October 24, 2017

”You never fail until you stop trying”
(Albert Einstein).

Abstract

This paper reviews three Artificial Intelligence (AI) algorithms: Case Based Reasoning (CBR), Genetic Algorithms (GA) and Support Vector Machines (SVM). Throughout this paper, we try to explain how and when to use them. To demonstrate that these algorithms don't solely have theoretical applications, some examples of existing solutions on the market are presented.

Contents

List of Figures

1 Introduction

Case Based Reasoning and Support Vector Machines are algorithms, developed to be used in Supervised Learning. Supervised Learning consists out of inferring a function from labelled training data (a set of training examples), which can be applied to map new examples. Genetic Algorithms were developed to be used in metaheuristics, which consists of a high-level procedure, created to find, generate or select a heuristic that provides a new optimal solution to an optimization problem. Case Based Reasoning (CBR) has its roots in the work of Roger Schank and his Yale students in the early 1980s with the model of dynamic memory, since which it has been used in help desk applications as well as health sciences among other domains. The original algorithm of a Support Vector Machine (SVM) was invented by Vladimir Vapnik and Alexey Chrevonenkis in 1963. It has been expanded in order to create non-linear classifiers by applying the kernel trick by Bernhard Boser, Isabelle Guyon and Vladimir Vapnik in 1992. Since then SVM has been used to help in face recognition and bioinformatics among other examples. Genetic Algorithms (GA) started as early as in 1954 but only became more common in the early 1960s when biologists started using it. Since then it began to develop more and more with the contributions of researchers like Fraser, Hans-Joachim Bremermann among others. This algorithm has been applied to areas like bioinformatics, economics, resolution of NP-Complete problems, etc.

This paper aims to present the three Artificial Intelligence algorithms in a such manner that a unexperienced user can grasp the concepts and usability of them. In order to do so, this paper starts by describing each algorithm, how they learn and what needs to be provided for them to function. Subsequently a few development tools that could be used to apply the algorithms, as well as some market solutions that are using the algorithms, are presented. In the last section a summary of CBR, SVM and GA with their respective differences can be found.

2 Case-Based Reasoning

Case-based reasoning (CBR) is the concept of solving problems by utilizing and revising solutions to similar problems from the past. Every case exists out of a problem description, a problem solution which states the derived solution to the problem, and an outcome which describes the state of the system after the case occured. All past cases are stored in a 'case base', where they are categorized in such fashion that similarities with any given case could be efficiantly found. For categorization, a vocabulary is used which contains the objects of interest.

2.1 The process

CBR can be divided into four consecutive steps: Retrieve, Reuse, Revise, and Retain. A representation of this process is illustrated in Figure 1.

1. Retrieve: After analysing the initial problem, CBR aims to find similar cases from the past that have been solved. By using similarity measure functions it computes the distance between the current and past cases. The past case with the smallest distance from the current one is selected and will serve as model to solve the current case.

2. Reuse: The selected retrieved case is adapted to fit the new case as much as possible. The solution to the adapted case subsequently will be used in an attempt to solve the current case.

3. Revise: The mapped solution from the retrieved case is evaluated and –if necessary– revised. Revising techniques aren't necessarily limited to the retrieved case nor CBR in general, and could include human interaction or other reasoning techniques.

4. Retain: When the proposed solution is being successfully adapted to the current problem, the current case will be stored in the case base and made available for future cases with similar problems.

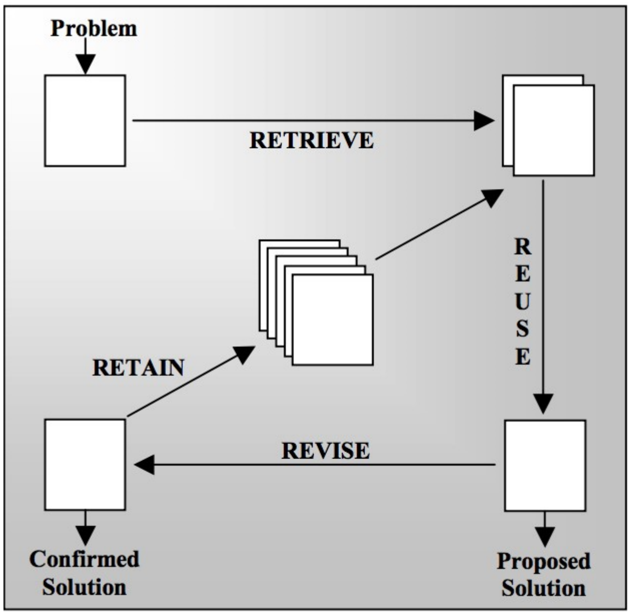


Figure 1: The CBR Cycle [1]

2.2 Assumptions

According to Kolodner[2], four assumptions represent the basis of the approach of CBR.

1. Regularity: Reoccuring actions in similar conditions tend to have similar outcomes.
2. Typicality: Situations tend to repeat themselves.
3. Consistency: When the situation has a little amount of changes, compared to past situations, the solution will probably have just a little amount of changes too.
4. Adaptability: Reocurring actions tend to have minimal differences. Those differences are assumed to be easy to compensate for.

2.3 Necessary domain knowledge

- Vocabulary includes the knowledge necessary for choosing the most relevant features, in such fashion that they are helpful in retrieving other cases with useful solutions to similar cases as well as being discriminative enough to prevent retrievel from too different cases.
- Similarity measures includes the knowledge to define the most appropriate case-retrievel in the used field.
- Adaptation knowledge could be described as the knowledge how differences in problems affect the solutions.

2.4 Attributes (features)

One of the necessities of CBR is the (supervised) selection of attributes. As CBR will use the attributes to calculate distances, choosing irrelevant attributes or out-of-proportion value ranges could decline the whole process.

2.5 Similarity measure function (SMF)

The term ‘distance’ between two cases could be defined as the euclidean distance in a k-d tree in which the amount of dimensions is defined by the amount of attributes. However, often prioritizing attributes (by giving weight) as well as non-linear behavior towards evalutation of certain attributes is wished for. Therefore, each attribute can have their own SMF –taking their distribution into account–, after which all local SMF’s are combined into a global SMF given a certain weight for each attribute.

2.6 Learning Capability

As a result of the learning approach, the outcome and competency of CBR is very much dependent on the used attributes and SMF. Even when provided millions of cases, the performance of CBR could be horrible when the retrievel step solely retrieves irrelevant cases. Therefore, frequently (re)evaluating the used attributes with their SMFs could improve the efficiency as well as the learning ability, especially within a context of frequent changes. Another aspect the learning ability greatly depends on is the mapping and revising of retrieved cases towards the new case. As CBR doesn’t specify the mapping nor revising, its learning ability greatly depends on the choices of implementation and therefore could not be generalized.

2.7 Anecdotal evidence

One of the main principles of CBR is the use of the nearest similar case. Even though the results could be great, only anecdotal evidence is used as a main operating principle. This means no statistically relevant data is used for backing the proposed solution. In other words, the retrieved case is implicitely being generalized.

2.8 Lazy generalization

In contrast to other learning techniques such as rule induction algorithms, CBR is not eager and uses lazy generalization. In other words, unlike for example neural networks, CBR doesn’t use propagation functions nor learning rules. This results in the great advantage of neglecting training time/computation as well as an explicit domain model. However, at query time -every time a new case is processed-, CBR needs to compute the relative distance to all existing cases, as well as adapt to the given problem. A possible way to increase calculation speed, is to regularily remove barely used cases.

2.9 Applications

In contrast to many algorithms that use deductive reasoning, CBR's approach of inductive reasoning allows it to be used with a minimum amount of data. This makes it very usable for situations where training data is unavailable or too scarce. The dispensability of a training phase makes it possible to instantly change behaviour of any of the local as well as the global SMF, as well as the content of the case base. Since its first use within commercial tools in the early 1990's, it has been used in a wide variety of domains:

- Diagnosis: Retrieved cases whose symptom lists are similar to the current one. The suggested diagnoses could be used from the best matching case(s).
- Help Desk: Customer service can handle problems related to products or services.
- Assessment: When variable values need to be determinated, past cases could help to find a good fit.
- Decision Support: When facing complex problems, searching for analogous problems in past cases could simplify the current problem. Especially document retrieval could be greatly simplified, hence the idea of keywords within documents.

Some examples of CBR based systems, with their respective application domain: Shrink (Psychiatry), Casey (Heart failure), Medic (Dyspnoea), Bolero (Pneumonia), Florence (Health care planning), Scina (Detection of coronary heart diseases), ICONS (Antibiotics therapy for intensive care), T-IDDM (Diabetes treatment), CADI (Cardiac ausculation diagnosis and instruction), and CAMP (Daily menu planning).[3]

2.10 Development tools

Even though there are plenty of software vendors as well as consultants who provide CBR solutions, two established open-source frameworks for developing CBR applications are:

- MyCBR: supports the developers in the construction of a knowledge model
- COLIBRI Studio: is focussed on the development of CBR applications As the focus might differ, it is not uncommon to combine the two frameworks in order to benefit from the strenghts of both.

3 Genetic Algorithms

The genetic Algorithm (GA) consists of a technique capable of optimizing the learning process based on the principles of genetics, Natural Selection and its basic principles were proposed by Holland. This system follows the same principle advocated by Charles Darwin, which reported the survival of the fittest, in the competition environment among individuals for scarce resources. That is, this technique allows us to find solutions to problems were the optimal solution is difficult to achieve and a good solution is enough, because this explores the various historical information and aim to eradicate the less viable solutions.

The solution that GA provides is represented as an individual of a population that is represented by a series of parameters, like the genes of a chromosome that can be represented as a array of values in a binary form.

3.1 The Process

In Figure 2, it is possible to observe the process of iteration of the genetic algorithm. It contains four basic operations:

- Calculation of fitness;
- Selection;
- Crossover;
- Mutation.

At the end of these operations, a new population with the most viable solution to the problem was generated. The initial population is generated by randomly assigning values to the genes of each chromosome. The gross fitness of a given individual of the population is measured through an error function, also called the objective optimization function of the problem. The raw fitness is then normalized (standardized fitness) to allow better control of the selection process. The criteria used to stop the algorithm tend to be that of the best individual's fitness together with the limitation of the number of generations.[10]

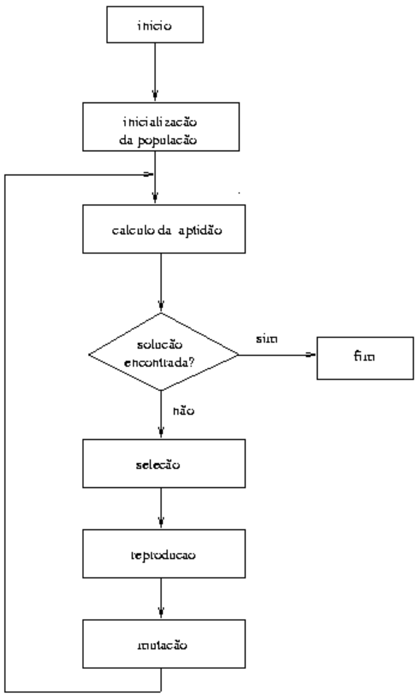


Figure 2: GA Process Diagram [10]

3.2 Learning Capability

In a practical application of a Genetic Algorithm, a population, which is a set of chromosomes, needs to be created and randomly initialized to ensure diversity and prevent premature convergence. A population is defined as a two-dimensional array of size X, that needs to be decided by a series of trial and error approaches as it shouldn't be too small nor too big. In terms of Population Models, there is the Steady state model which generates one or two children in each iteration of the algorithm which replaces one or two individuals, and the Generational model which generates as many children as individual in the population and replaces the previous population completely by the newest one every iteration.[8]

GA learning methods involve selecting two chromosomes, called "parents", to recombine their genes in order to produce new children for the next generation through some sort of parent selection technique.[9] This part of the algorithm is very important as better parents produce better and fitter children which results in a good convergence rate [11]

A fitness proportionate selection model means that every individual is assigned a probability that is proportional to the fitness of becoming a parent. Both implementations could be represented by a wheel (pie chart) divided in n parts, where n is the number of individuals in the population [11]. It should be noted that this technique doesn’t work with cases where the fitness can have negative values.

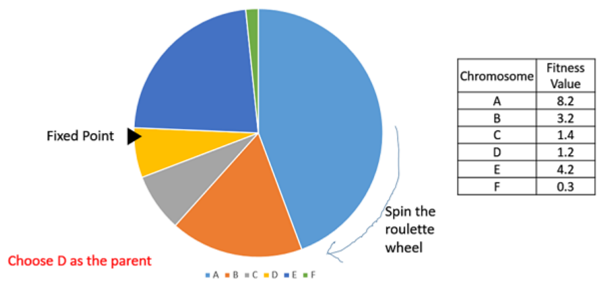
In Roulette Wheel Solution, we define a fixed point and rotate the wheel. The region that is selected is chosen to be the parent. This is used to select both parents. How an individual is chosen depends only on its fitness because the fitter the individual is the bigger the pie on the wheel. [9][11]

In terms of code it could be implemented following these steps, Figure 3 [9]:

1. Sum the fitness of the whole population and store it in a variable, for example, totalFitness;

2. Then a random number between 0 and totalFitness is generated;

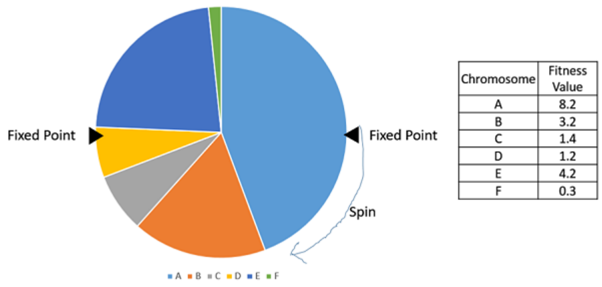
3. Return the first individual whose fitness added to the partial sum of the fitness of previous members of the population is superior or equal to totalFitness;



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figure 3: Roulette Wheel Selection [12]

The Stochastic Universal Sampling method is similar to the method presented before, but instead of having just one fixed point, there are multiple fixed points so both parents can be chosen in just one spin. This allows the fitter individuals to be chosen at least once, Figure 4.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figure 4: Stochastic Universal Sampling [12]

When parent selection techniques were explained, the concept of fitness associated with each individual of the population was introduced. This concept represents how “good” a solution is when considering the problem, which is calculated with a function that takes a candidate solution as input and produces the fitness value as output.[13] It should be noted that this function is called repeatedly and therefore should be fast.

In terms of how the new individual are generated, the crossover operator and the mutation operator need to be applied.

The Crossover Operator, is applied in Genetic Algorithm with a high probability, and is applied to the parent chromosomes. [14] One point Crossover is a mechanism where a random crossover point of the chromosome is chosen to be swapped. The portions of the two chromosomes on the right side of this point are swapped in order to generate the children, Figure 5 [9].

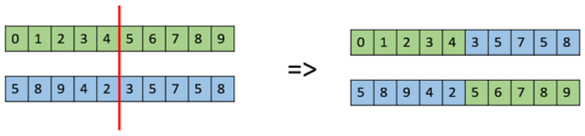


Figure 5: One point Crossover [12]

Uniform Crossover is an operator that unlike the One Point crossover treats each gene separately. In this operator we “flip a coin” to decide if the gene is going to be part of the children or not. This allows some bias as we can make a child have more genetic material from a specific parent than from the other,Figure 6.[14]

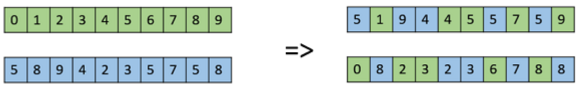


Figure 6: Uniform Crossover [12]

Mutation operators are applied to the new individual rather than to the parents and consists in a random change of their chromosome so it guarantees a new solution.[9] It should be noted that this type of operator is applied in Genetic Algorithm with a lower probability. Bit Flip Mutation is mostly used in binary coded Genetic Algorithms. To apply this method one or more genes (in this case bits) are chosen and inverted, Figure 7. [15]

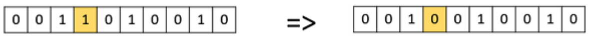


Figure 7: Bit Flip Mutation [12]

Swap Mutation is mostly used in permutation coding because two genes of the chromosome are chosen randomly and switched position [15].

The cycle of “evolution” occurs until a termination condition has been reached. This termination condition should be such that the final solution is close to the optimum. To achieve that, we can use criteria such as defining a predefined number of computational runs or when there hasn’t been any improvement of the population in X iterations [9].

3.3 Advantages and Limitations

The use of Genetic Algorithms brings some advantage over the “traditional” methods. GA provides more than one good solution and always provides a solution that can get better over time as the population gets fitter [16]. It has good parallel skills and is suitable for distributed computing which is useful as GA can be a good option when the problem has many parameters and derivative methods can’t be used. [17]

This algorithm is very useful when it comes to resolving stochastic problems, optimization problems like creating a schedule for schools and universities (that is as NP-hard problem), where many constraints have to be satisfied and an optimal solution can never be found but a good solution might fit. [18] Those are some of the most relevant advantages in GAs.

On the other hand, like all the algorithms GAs have disadvantages. The biggest limitation of this algorithm is that it can’t guarantee the optimal/global solution as it uses stochastic methods and can get stuck at a local optimal solution in an early stage, for example, because of a lack of diversity that allows the population to be taken over by one extremely fit solution that lead to a premature convergence. This algorithm is also computationally expensive in terms of calculation of the fitness value and in terms of time because it might need to run through various generations to start demonstrating good results.

Those are some of the most relevant limitations in GAs. Other limitations include the trial and error nature of deciding the size of the population, deciding the most useful fitness function, genetic encoding and some others.[19]

3.4 Development Tools

There are many development tools for Genetic Algorithms also known as Evolutionary Computation that can be used with programming languages such as JAVA, C++, Python, etc. Open BEAGLE is a C++ Evolutionary Computation open source framework developed by Google that provides a software environment that is generic, portable, efficient, robust and user friendly. This framework is an architecture that follows the Object Oriented programming principle, on which the generic EC frameworks are built . [20] Some of the features that this framework supports are evolution strategy, integer-valued vectors and real-valued vectors, master-slave model for parallel fitness evaluation, history of the best-of-run individuals for the whole population and each demes (vivarium), and the hierarchical division of the population in four levels: vivarium, deme, individuals and genotypes.[21]

. JCLEC also known as Java Class Library for Evolutionary Computation is an open source software system for Evolutionary Computation research and provides a framework for GA, genetic programming and evolutionary programming. Some of the GA features of this framework are that it allows linear binary encoding, linear integer encoding and linear real encoding for the operations of Crossover and Mutation.[22] Additionally, it also provides some general features like multi-threading, extensibility, generality, reusability and multiple random numbers generators.[?] This framework provides some GAs examples of how to approach solutions to certain NP-Hard problems like the Knapsack problem using binary encoding, the Travelling Salesman Problem using integer encoding and also how to optimize real functions using real encoding.[24]

3.5 Applications

Genetic algorithms have a variety of applications in the field of Artificial Intelligence as it's a search method that needs little information to search efficiently in a big or poorly-understood search space. [16] GAs can be used for encryption of data and code breaking, for example, if someone codifies an encryption algorithm someone else can create a GA that could break that encryption after being trained [17].

Specialized GAs can be used in order to present some good solutions to The travelling salesman problem, in which a valid solution represents a route where each location is visited only once and the mutation and crossover methods need special requirements (for example: the mutation method cannot add nor remove a location only shuffle the existing route). The crossover method needs to guarantee that the offspring presents every location. [18]

Another example of a GA field of application is neural networks since both of them show a high problem-solving ability they can be combined and a GA will be used to optimize the network topology, learning rate, etc. Combining GA and NN consists of encoding in the genes the information about the NN and then generate a population and create a NN according the information mentioned above to evaluate the parameters. The following procedure depends on the type of GANN strategy that is being followed but it normally is to determine the performance with back-propagation training and then use fitness evaluation to rank those performances. The final step is to apply mutation and crossover operations to create new offsprings that will replace the weaker individuals of the population. [32]

Genetics Algorithms are also used in Bioinformatics, for example the work of Chuzhanova used GA combined with the Gamma test to find feature subsets in order to classify large subunits of RNA as mentioned in “A review of feature selection techniques in bioinformatics” by Yvan Saeys, Iñaki Inza and Larrañga Pedro. [33] In this field of application, the paper “GARD: a genetic algorithm for recombination detection” by Pond Sergei, Posada David,Gravenor Michael, Woelk Christopher and Frost Simon shows how GA was used to develop a similarity-base model selection procedure to detect if there is any sign of recombination in alignments of DNA sequences. [34]

Furthermore, there are various scientific papers or books with applications of GA that can be consulted, such as: ”Web Information Retrieval Using Genetic Algorithm-Particle Swarm Optimization” [35], “Pratical Genetic Algorithms” [36], “Genetic Algortithms and Engineering Optimization”. [37]

4 Support Vector Machines

Support Vector Machines (SVMs) is a technique used for classification, regression and outliers detection. [48] However, it is mostly used in classification problems. SVM consists of a set of supervised learning methods having learning algorithms associated to a specific model which is created by the analysis of a training data, a set of training labelled examples marked as being part of a category or more, makes it possible for the model to assign new examples to one category or another. [39][49] SVM is already being performed in real-world applications like classification, recognition, classification is one of the standard tools for Machine Learning and Data Mining. [38]

In the SVM algorithms each item of the training dataset is represented as a point in a n-dimensional space where each dimension coordinate is associated to a feature, the space has as much dimensions as numbers of features.[39] All the points represented in a coordinate are called Support Vectors, with which a hyperplane is calculated, being this the frontier which best segregates the different classes. The points represented correspond to individual observations, for example, the SV (20,33) [green SV] corresponds to a dog and the SV (60, 89) [red SV] corresponds to a cat in a classifier model where we have the two classes dog and cat.

In the example, as found in Figure 8, only two features are being represented. However, often multiple features can be identified, creating high dimensional feature spaces. While the case above was a Linear Problem, this could often result in a Non-Linear Problem.

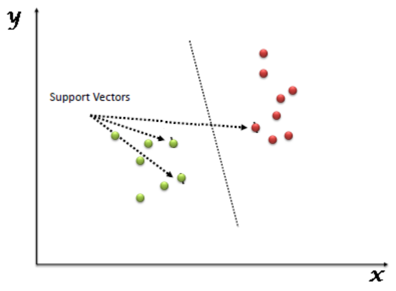


Figure 8: Dimensional representation of a "two classes problem" [55]

Nello Cristianini [38] explains that SVM is a class of kernel functions (notice that kernel functions have not in common with kernel of an OS). A similarity function (kernel function), which is provided to the SVM, “takes two inputs and spits out how similar they are” [49]. The kernel functions allow us to efficiently process these n-dimension spaces (where n bigger than 2) that could have a high number of features. A small example of the application of kernel functions to find the hyper-plane in the case represented in Figure 9 is represented in Figure 10.

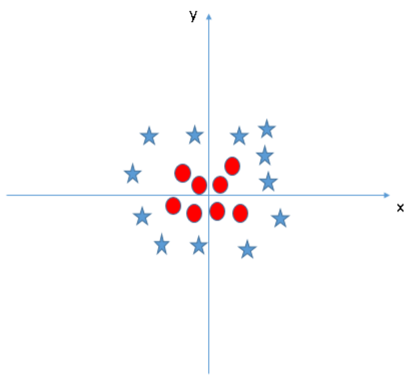


Figure 9: Non-linear problem [55]

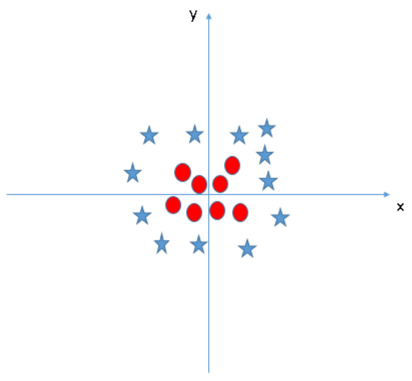


Figure 10: Non-linear hyper-plane [55]

However, as Nello Cristianini explains in [38] “The danger of overfitting inherent in high dimensions requires a sophisticated learning bias provided by the statistical learning theory”, there is the risk of overfitting associated to the usage of kernel functions. Overfitting, as the name suggests, is when a model is excessively adapted to a specific training dataset, having great results with it but mediocre results with new data.

Like everything, SVMs has some pros and cons [40]. Some of the SVM pros are: [38] it works really well when there exists a clear margin of separation between SVs;[39] efficiency in high dimensional spaces; [40] efficiency in cases where number of dimensions is greater than the number of samples; and [41] memory efficiency. On the other hand, SVMs represents a range of cons: [38] low performance with large datasets because of long training time; and [39] low performance with high noise.

4.1 Learning capability

In a practical application, a Support Vector Machine algorithm learns a linear model, in result of the training of the SVM with labelled data, which allows it to make classification. [53] A linear model is a hyperplane (a subspace of one dimension less than its ambient space) that will separate the data and categorize it. [52] As the hyperplane can have various possible types of representations, a conventional one as chosen: the canonical hyperplane represented by —b+mx—=1, where the x symbolizes the support vectors.

The first step to obtain a linear model is to segregate the two classes to find the right hyperplane, supposing that the data can be separated by it, using a set of mathematical functions, known as kernels. The second step is to choose the right hyperplane because if it passes too close to the points that represent the data it can be noise sensitive and therefore will not generalize correctly. There can be two ways to find it: or the distance (known as margin) between the nearest data point, from either class, where the hyperplane is maximized and the one that has the highest distance is chosen [52], or the line is represented by the equation y = mx + b and an algorithm is used to determine what are the values of m and b that produce the "optimal" hyperplane. [53]

Support Vector Machines can also work with non-separable data and therefore learn a nonlinear model. When data cannot be clearly separated in the low-dimensional space, like R^2, a technique called the kernel trick is used to find a transformation to a higher dimensional space, like R^2 to R^3 such that the data can be linearly separable in R^3. The Kernel Trick basically takes a dataset D that cannot be linearly separable in R^N but maybe linearly separable in a higher dimensional space R^M and transforms it in a dataset D'. [54] As D' is linearly separable the SVM can be trained to use a linear model to find a hyperplane that separates the classes in D', this allows to use both nonlinear models and linear ones, [54] Figure 12 and Figure 11.

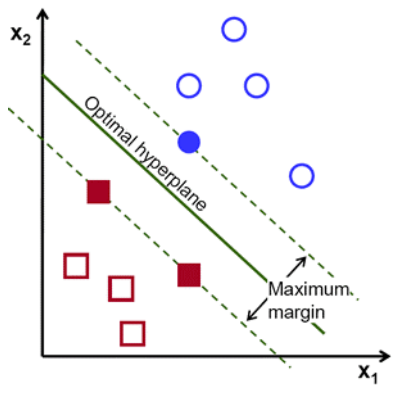


Figure 11: Finding the optimal hyperplane by maximizing the margin [56]

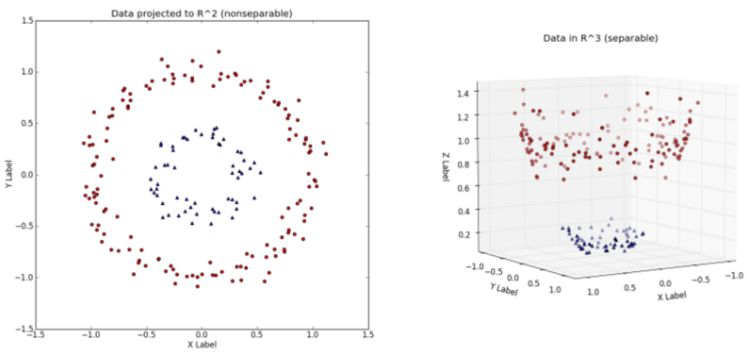


Figure 12: Hyperplane that segregates the data correctly [56]

4.2 Development tools

Nowadays there are tools for almost everything. Because of those tools we don't have to define all the structure of a SVM such as the mathematics behind it or the kernel functions. In the main programming languages used for Machine Learning powerful tools are available to make the use of SVM easier. Using these libraries or frameworks the user just has to implement and parameterize the SVM in order to create the most suitable model to his or her needs.

Some of the most powerful programming languages we have for SVM (and Machine Learning) are for example Python, R, MatLab and more recently Java. Each one of these technologies have a range of mature and tested open-licensed frameworks/libraries.

An example of a powerful framework for Python of Machine Learning in general, including SVM, is the Scikit-learn [41] which is built on NumPy, SciPy and matplotlib. Scikit-learn can be used for Classification, Regression, Clustering and much more. Equivalent tools with R can be found like E1071 [42], a set of functions for multiple purposes by the DSPTG, or LIBSVM [43] which is a specific library for SVMs implementation. We can even find frameworks with multiple language compatibility like the famous OpenCV [44, 45] available in the C family, Python and Java, or Google's framework TensorFlow - available in Python, C++, Java and the most recently trendy programming language Go.

4.3 Applications

One of the books used as base of knowledge is "An Introduction to Support Vector Machines and other kernel-based learning methods" by Nello Cristianini, John Shawe-Taylor [38], two specialists in the fields of Artificial Intelligence and Statistics, refers to SVM as a solution for different problems and a few more are indicated by other authors.

SVM is now being used in different applications. Examples of SVM applications are: [38] Face Detection where SVMs classify parts of an image as a face and non-face and create a square boundary around the face; [39] Text and hypertext categorization (THC) where SVMs allow THC for both inductive (reasoning from training cases to test cases) and transductive models (reasoning training cases to general rules and applying them into test cases); [40] Classification of Images where SVMs provides better search accuracy for image classification then the traditional query based searching techniques; [41] Bioinformatics for example on protein, cancer, genes and patients classification; [42] Handwriting recognition.

In addition to the above SVM applications, a wide list of scientific papers of SVM applications can be found [51], such as: "Dynamic Reconstruction of Chaotic Systems from Inter-spike Intervals Using Least Squares Support Vector Machines"; [39] "Application of The Kernel Method to the Inverse Geosounding Problem"; [40] "SVM for Geo- and Environmental Sciences"; [41] "SVM for Protein Fold and Remote Homology Detection".

5 Conclusion

In the field of artificial intelligence, many (sub)methods can be used to solve a given problem. Even when some might offer similar results, computation/implementation time can differ greatly. We discussed three different methods that could be used to propose (a part of) a solution for a given problem. We discussed how Case-Based Reasoning (CBR) can be used to solve problems by using aquired knowledge from previous experiences. In order to do so, past cases are being stored and proposed (and possibly adapted) when a similar problem occurs. For this method, it is necessary to have a certain knowledge about the domain, and specify its features with associated similarity measures. A second topic this paper discussed is Genetic Algorithms (GA), which is a method for solving optimization problems and is based on the natural selection process within biologic evolution. At the start, a group of possible solutions ('the population') is initiated with certain (of pseudorandom) values. In every iteration -in this context known as generation- a selection happens where usually (the fittest) part of the population is kept while the rest of the population is replaced with modified (recombined, sometimes mutated) individuals. The selection of fitness happens through a fitness function. After a certain amount of iterations, the 'fittest' individual will most likely contain the most ideal (set of) sought-after variables. This method is especially useful when it is not difficult to make a genetic representation of the solution domain as well as the fitness function, but difficult to find possible solutions. The third en last method we've discussed is Support Vector Machine (SVM), a supervised learning model that can be used for classification and regression analysis. This model requests a big amount of training data but doesn't necessarily require knowledge of the solution domain. It aims to find the most suitable space (one or more hyperplanes) to separate the given data into two sets. SVM tries to maximize the distance between the data points from both sets. This could be achieved by training the SVM with labeled data points (input of which the classification is already known). Later on, its performance could be verified by using another set of labeled data points. Considering the variety of problems that could occur, all three discussed methods could be beneficial in certain contexts. Being aware of the different options to solve problems within artificial intelligence could greatly improve development time as well as performance. Even though this paper only touches a tiny part of the possibilities regarding problem solving, by comparing three very diverse methods it becomes clear that a properly chosen method could provide huge benefits towards finding proper solutions.

References

[1] https://ibug.doc.ic.ac.uk/media/uploads/documents/courses/syllabus-CBR.pdf, 21 October 2017

[2] J. Kolodner, “Making the implicit explicit: Clarifying the principles of case-based reasoning”, Case-Based Reasoning: Experiences, Lessons & Future Directions, D.B. Leake, (Ed.), pp. 349-370, AAAI Press, Menlo Park, USA, 1996, 21 October 2017

[3] https://www.researchgate.net/publication/220254409_Medical_applications_in_case-based_reasoning, 23 October 2017;

[4] ttp://artint.info/html/ArtInt_190.html, 21 October 2017

[5] ttps://ibug.doc.ic.ac.uk/media/uploads/documents/courses/syllabus-CBR.pdf, 21 October 2017

[6] ttp://www.expertupdate.org/papers/13-1/CRC%20final%20version%20UKCBR2012-Roth-Berghofer%20et%20al.pdf, 21 October 2017

[7] ttp://mycbr-project.net/downloads/myCBR_3_tutoria_slides.pdf, 21 October 2017

[8] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_population.htm , 14 October 2017

[9] Tang K.s. , Kwong Sam, “Genetic Algorithms: Concepts and applications” , City University of Hong Kong, 1996

[10] http://www.nce.ufrj.br/GINAPE/VIDA/alggenet.htm

[11] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm , 14 October 2017

[12] Tutorials Point, Genetic Algorithms, Parent Selection, 14 October 2017

[13] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_fitness_function.htm , 14 October 2017

[14] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm , 14 October 2017

[15] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm , 14 October 2017

[16] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm , 14 October 2017

[17] https://stats.stackexchange.com/questions/23106/benefits-of-using-genetic-algorithm , 18 October 2017

[18] https://watchmaker.uncommons.org/manual/ch01s02.html , 20 October

[19] https://www.quora.com/What-are-the-disadvantage-of-genetic-algorithm , 18 October 2017

[20] https://code.google.com/archive/p/beagle/ , 20 October 2017

[21] Gagné Christian, Parizeau Marc, "Open BEAGLE: A new versatile C++ framework for evolutionary computations", Laval University, 2002

[22] http://jclec.sourceforge.net/ , 20 October 2017

[23] http://jclec.sourceforge.net/index.php?option=com_content&view=article&id=2&Itemid=2 , 20 October 2017

[24] http://jclec.sourceforge.net/mediawiki/index.php/Examples , 20 October 2017

[25] https://www.doc.ic.ac.uk/ nd/surprise_96/journal/vol1/tcw2/article1.html , 19 October 2017

[26] https://www.brainz.org/15-real-world-applications-genetic-algorithms/ , 15 October 2017

[27] http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5 , 15 October 2017

[28] Joehn Philipp, "Combining Genetic Algorithms and Neural Networks: The Encoding Problem.", The University of Tennessee, 1994;

[29] https://www.doc.ic.ac.uk/ nd/surprise_96/journal/vol1/tcw2/article1.html , 19 October 2017

[30] https://www.brainz.org/15-real-world-applications-genetic-algorithms/ , 15 October 2017

[31] http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5 , 15 October 2017

[32] Joehn Philipp, "Combining Genetic Algorithms and Neural Networks: The Encoding Problem.", The University of Tennessee, 1994;

[33] Yvan Saeys, Iñaki Inza and Predro Larrañga, "A review of feature selection techniques in bioinformatics",2007;

[34] Pond Sergei, Posada David,Gravenor Michael, Woelk Christopher and Frost Simon, "GARD: a genetic algorithm for recombination detection" ;

[35] Borkar Pryia,Patil Leena,"Web Information Retrieval Using Genetic Algorithm-Particle Swarm Optimization", December 2013

[36] Haupt Randy, Haupt Sue,“Pratical Genetic Algorithms”;

[37] Gen Mitsuo,Cheng Runwei,“Genetic Algortithms and Engineering Optimization”;

[38] Nello Cristianini, John Shawe-Taylor, “An Introduction to Support Vector Machines and other kernel-based learning methods”, Cambridge University Press, 2000;

[39] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/, 14 October 2017;

[40] https://www.analyticsvidhya.com/blog/2014/10/support-vector-machine-simplified/, 14 October 2017;

[41] http://scikit-learn.org/, 14 October 2017;

[42] https://CRAN.R-project.org/package=e1071, 14 October 2017;

[43] https://www.csie.ntu.edu.tw/ cjlin/libsvm/, 14 October 2017;

[44] https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html, 14 October 2017;

[45] https://opencv.org, 14 October 2017;

[46] Martin Law, “A simple Introduction to Support Vector Machines”, https://www.cise.ufl.edu/class/cis4930sp11dtm/notes/intro_svm_new.pdf, 14 October 2017

[47] http://www.datasciencecentral.com/profiles/blogs/real-life-applications-of-support-vector-machines, 14 October 2017

[48] http://scikit-learn.org/stable/modules/svm.html, 19 October 2017

[49] https://en.wikipedia.org/wiki/Support_vector_machine, 19 October 2017

[50] https://www.quora.com/What-are-kernels-in-machine-learning-and-SVM-and-why-do-we-need-them, 19 October 2017

[51] http://www.clopinet.com/SVM.applications.html, 19 October 2017

[52] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ , 20 October 2017

[53] https://www.svm-tutorial.com/2017/02/svms-overview-support-vector-machines/ ,20 October 2017

[54] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html , 21 October 2017

[55] Analytics Vidhya, SVM simplified

[56] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html