Detecting SARS and Other Medical Conditions using CASPER and a Custom Evolutionary Algorithm

Michael Polkinghorne

Research School of Computer Science, Australian National University
u5845781@anu.edu.au

Abstract. This paper investigated an artificial dataset of patients' medical observations and diagnoses with the aim of producing a robust classifier that could handle noisy measurements. This classifier was a CASPER neural network whose hyperparameters were tuned using a bespoke evolutionary algorithm. To force robust classification, it was trained with a substantial degree of noise added to the dataset ($\sigma = 60\%$ of range), and it was able to achieve an MSE of 0.036 as well as perfect classification on the original test data. In addition to the classifier, this process provided a set of good hyperparameters for CASPER, however some of these differed substantially from the parameters suggested by CASPER's authors. It was unclear whether these hyperparameters were really optimal, or whether they were one of many acceptable combinations that would work well on the simple classification task considered. This makes it unlikely that they would generalise well to other tasks. The evolutionary algorithm itself appeared to work reasonably well, however it was very difficult to evaluate its performance due to computational resource limits.

Keywords: CASPER, evolutionary algorithm, neural network, hyperparameter optimisation, classification, SARS.

1 Introduction

This paper aims to analyse a dataset [3] of medical observations about patients such as temperature and blood pressure, and used them to develop a robust, generalisable model for the purpose of hypothesising diagnoses about yet-unseen patients. Although SARS is no longer a common diagnosis and the dataset only includes a small number of medical conditions, developing a similar classifier on an extended dataset with more medical conditions could potentially be very useful for assisting doctors in diagnosing patients and quickly alerting them of patients likely to have dangerous or contagious medical conditions.

The particular dataset [3] analysed considered normal patients, and patients with three different medical conditions: *high blood pressure*, *pneumonia*, and *SARS*. Patients are either *normal* or have one of the specified conditions, so no compound conditions were present. There were a total of 4000 patients, with 1000 in each of the four categories.

The features in the dataset include a variety of observations and basic measurements about the patients. These are: *temperature*, *(degree of) nausea*, *blood pressure*, and *(presence of) abdominal pain*. The temperature measurements were taken at four different times of day: 0800, 1200, 1600, and 2000. The blood pressure measurements included separate systolic and diastolic values. Since the paper [3] aimed to classify the patients using fuzzy logic methods, each measurement was pre-processed into either two or three labels, with the features taking the form of a fuzzy membership value on [0, 1] for each label. The labels are presented in *table 1*.

Table 1: measurements and fuzzy labels in the data [3]

Measurement	**Associated fuzzy labels**
Temperatures (0800, 1200, 1600, 2000 hours)	'Slight', 'Moderate', 'High'
Nausea (single measurement)	'Slight', 'Medium', 'High'
Blood pressure (systolic and diastolic)	'Slight', 'Medium', 'High'
Abdominal pain	'No', 'Yes'

Fuzzy sets are a method of data classification initially proposed by Zadeth [6]. They are analogous to conventional sets in the sense that they are collections of elements, but they capture degrees of membership by assigning a membership value on [0, 1] [6]. In the context of this dataset [3], the fuzzy sets are used as a replacement for exact measurements, in order to allow the data to be processed using fuzzy logic methods.

Since this paper uses a neural network algorithm, CASPER, the use of fuzzy labels is unnecessary, and it was of interest to be able to convert the fuzzy labels back to measurements, normalised or otherwise; however CASPER is also capable of using the fuzzy labels as input directly. It is therefore important to understand how the data were generated.

It was stated [3] that the data were artificially generated, but the process used was not explained. Furthermore, it is unclear whether the data were directly generated as fuzzy labels or whether they were generated in another form and converted. The fuzzy signature used by [3] to classify them was originally designed in [4]. [4] listed what labels were included but did not include numerical values for them. The distributions of the data were thus analysed manually to determine the numerical meanings of the labels.

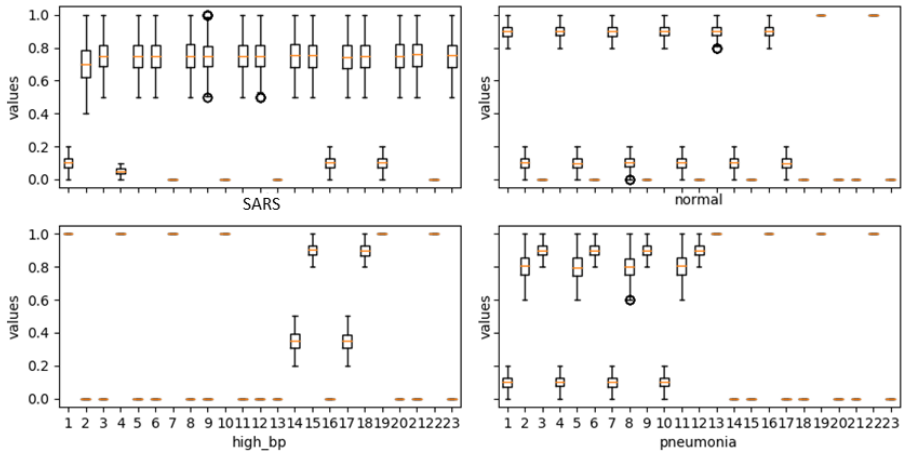

Figure 1: boxplot of each of the labels in the data. All labels are in the order shown in table 1

By analysis of the *normal* data, it seems that patients usually had a 'slight' temperature at all four times measured, and had a small membership in the 'moderate' category. This suggests that the labels indicate a temperature range where 'slight' is at or near a normal temperature for a patient, rather than an indication of a measured abnormal condition as the label would appear to indicate. Similarly, a 'slight' blood pressure appears to be a nominal measurement. In all cases, all patients were strongly members of at least one label.

As a result, it was assumed that a non-fuzzy representation could be constructed by defining the labels to represent uniformly spaced points with the smallest being 0 and the largest being 1. This assumption allowed a real value on [0, 1] to be generated by taking the membership-weighted average of the labels. The defuzzified data are shown in *figure 2*.

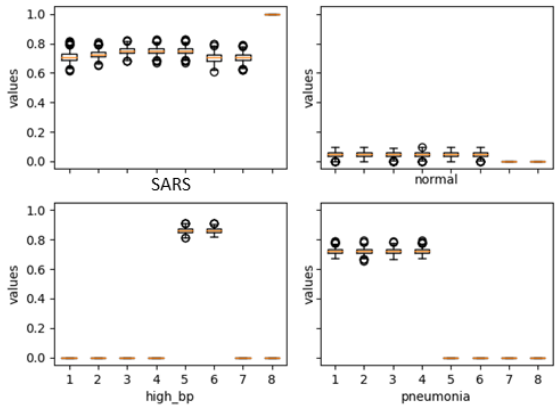

Figure 2: defuzzified representation of the medical dataset. Values are presented in the order corresponding to table 1

Upon inspection of both the fuzzy and defuzzified representations, it can be seen that the data in each category have a very small variance on each feature and that there are no apparent outliers. Furthermore, for any chosen pair of categories, several distinguishing features can be found such that there is no overlap between the values. Thus it is expected that the data are easily separable. This is confirmed in the paper [3], which was able to completely separate it using cascaded fuzzy signatures.

2 Method

2.1 Applying CASPER to the data

Since the data features were fuzzy memberships in the range [0, 1], they were already in a suitable form to directly input into CASPER networks. The features were not rescaled to a common variance because the ranges already have a semantic meaning – the membership strength in the fuzzy sets. Rescaling in this manner would artificially inflate the importance

of small differences between values, and encourage CASPER to give too much consideration to them, resulting in a less robust model.

In the defuzzified data, the ranges were already rescaled to [0, 1], so no further rescaling was necessary.

2.1 Linear separation of the data

The original dataset paper [4] was able to correctly classify all points in the data using hierarchical fuzzy signatures. Since the dataset was, by observation, linearly separable, it was expected that CASPER should be able to reproduce this correct classification without needing hidden neurons at all.

The preliminary aim of the paper was to reproduce the classification result obtained on the dataset in [4] with fuzzy signatures. [4] was able to correctly classify all points.

For this task, the CASPER algorithm was tested with a length of zero, meaning that no hidden neurons were present, and all other hyperparameters as per *table 2*. This was repeated for both the fuzzy and defuzzified representations of the features. Up to 15 iterations were tested. The hyperparameters were set as per *table 2*. Where they are listed as *varied*, the value from the CASPER paper was used.

2.2 Data modifications to ensure robust fitting

Since the data in the SARS dataset were already very well separated, an infinite number of linear classification boundaries were possible, and many of them could completely ignore some of the features. On this dataset, all such classifiers would be equally effective, however some would clearly produce better results than others in practise.

In reality, there are many more than four classes of medical conditions to consider, and patients may present with more complex combinations of medical conditions. Additionally, some measurements may be taken inaccurately. I therefore define a good model to be one that has a good chance of effectively generalising to real-world measurements that were not specifically generated for a classification problem.

Since additional medical datasets were unavailable for this task, I aimed to make the model as robust as possible with only this dataset, and to develop a method that could generate more robust predictions when trained on other datasets in the process. A robust classifier should: use all relevant information to make a prediction rather than just a small number of features; produce classification boundaries that are far from the means of the features in each sample; and continue to produce sensible results if noisy or inaccurate measurements are taken.

In order to achieve robust classification, a large degree of noise was added to the dataset, thus forcing any classifier trained on the dataset to be robust in order to achieve a good result. One significant problem with this approach is that adding noise removes information about the data. For example, a point on the extreme edge of a feature for a certain condition may be randomly moved back towards the centre of a cluster by the added noise to it. In order to reduce this issue and further reduce overfitting, three copies of the data were made, and independently sampled noise was added to each copy. In order to avoid leaking information about the test data into the training set, the data were split into the sets prior to the repetition and noise-adding operations. The variance of 0.6 was chosen so that no configuration would produce an MSE close to zero, thus ensuring that the MSE gradient would not vanish and that there would be a suitable error function for selecting the best model and parameters.

The procedure for this was as follows:

- Split the data into training and test sets
- For each set:
    - Repeat each record three times
    - Add an independently-drawn value from a Gaussian distribution with a mean of 0 and a variance of 0.6 to each point
- Proceed with training as normal on the new dataset.

This testing was done only with the fuzzy data, since the data was fuzzy.

2.3 CASPER Algorithm and hyperparameters

This dataset was classified using the CASPER algorithm [1], a form of neural network where neurons are sequentially added and trained as necessary in order to predict the provided target labels with sufficient accuracy [1]. These properties make it an attractive choice of algorithm for predicting the features.

Cascade correlation algorithms, including CASPER, are neural network algorithms which begin with just an input layer and an output layer, and no hidden neurons. After initial training, they grow and improve the network by adding and

training new hidden neurons sequentially. Algorithms in this family differ by how they add and train the neurons, and this can make a large difference to the quality of the results. [2]

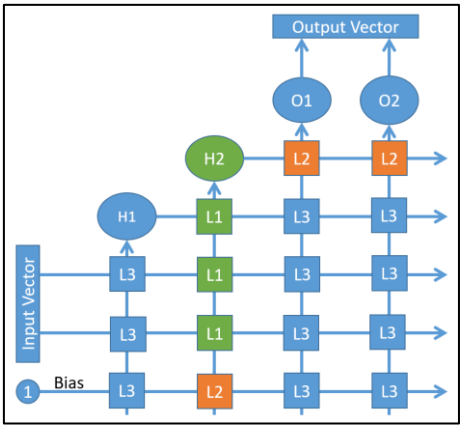

Figure 3: structure of CASPER with learning rates. Based on a diagram in [1].

CASPER is one of the most successful algorithms in this family [2], and it is distinguished by training the whole network simultaneously using variable learning rates and the RPROP algorithm [1]. The motivation for CASPER was to retain the beneficial results from CasCor, whilst avoiding the problems of overfitting associated with freezing training of all of the other neurons [1]. This is accomplished by setting different learning rates for different weights within the network. All weights of the new neuron's inputs (except the bias) are at the highest rate L1, the weights between the new neuron and the output neuron are at an intermediate rate L2, and all other weights are at the lowest rate L3 [1]. In the original CASPER paper [1], these are 0.2, 0.005, and 0.001 respectively.

For optimisation, CASPER uses a modified version of RPROP [1]. RPROP is a replacement for gradient descent where the step size is not proportional to the gradient [5]. Rather, on each feature, the algorithm steps in the opposite direction to the gradient with a step size Δ. When the gradient stays the same for two consecutive steps, Δ is increased according to $\Delta_{t+1} = \Delta_t \times \eta^+$, and when the gradient changes signs, no step is taken and delta is decreased according to $\Delta_{t+1} = \Delta_t \times \eta^-$. Additionally, bounds are placed on Δ for numerical stability. [5]

The three learning rates L1, L2, and L3 in CASPER are the initial values for Δ for the relevant connections, thus functioning as initial learning rates. Since RPROP steps can grow exponentially, the learning rates of the L2 and L3 connections can eventually reach or exceed the L1 learning rate, to the function of the different weights is to force the new neuron to be trained before other neurons can be significantly changed. [1]

As a cascade correlation algorithm, CasPer is very sensitive to the choice of hyperparameters [2], thus optimising them for the specific problem is imperative, and a significant focus of this paper.

Although CASPER's dynamic cascade architecture allows it to grow as needed to fit the problem at hand, it requires many hyperparameters to be appropriately tuned, some of which are highly problem-specific [1]. Furthermore, some of the general parameters may be better tuned for the specific problem. As a result, much of the focus of this paper is tuning CASPER's hyperparameters.

Table 2: CASPER's suggested [1] hyperparameters, and the hyperparameters used in this paper.

	Name/s	**Value**	**Value in CASPER paper [1]**	**Meaning**
General neural network parameters	N_{hidden}, length	*Varied*	None suggested / problem-specific	Number of hidden neurons / length of hidden neuron chain
	Activation function	Sigmoid (0,1)	Problem-specific	The function used to make neuron outputs non-linear and constrain their range
	Addition to derivative	0.0001	0.0001	A small constant added to the sigmoid's derivative to prevent neurons getting 'stuck'
	σ_{init}, noise_std	0.002	N/A	Variance of Gaussian random noise added to new connection weights.
Stop conditions	Minimum required decrease	Not used	1%	Amount that the MSE must fall per epoch for training to continue.
	N_{iter}	*Varied*	Not used	Amount of iterations after which to finish training each neuron.

CASPER-specific learning rates	L1	*Varied*	0.2	Initial learning rate for the new neuron (and the output layer before hidden neurons are added)
	L2	*Varied*	0.005	Initial learning rate for the output neurons' weight for the new neuron
	L3	*Varied*	0.001	Initial learning rate for the new neuron's bias, and for all other neurons' weights.
Normalisation	k	*Varied*	Problem-specific	Normalisation constant used to regularise the weights
Additional RPROP parameters	η^-	0.5	0.5	How much to multiplicatively reduce delta (step size) when the gradient changes sign
	η^+	1.2	1.2	How much to multiplicatively increase delta when the gradient's sign remains constant and nonzero
	Δ_{min}	1e-6	1e-6	Minimum allowed value for delta
	Δ_{max}	50	50	Maximum allowed value for delta

2.4 Genetic algorithm and hyperparameters

For this task, we aim to use an algorithm to optimise the hyperparameters of CASPER.

Nature of this problem

The CASPER hyperparameter optimisation problem is a difficult one. First, CASPER has a degree of randomness when initialising the weights for new connections, thus the final weights and output of the network are stochastic, even for fixed input data. There is no convenient method of computing the gradient of the output, except empirically through multiple trials over a small region of the search space.

Furthemore, CASPER itself uses RPROP (a form of gradient-based descent) so the network will generally produce weights at (or very close to) a local minimum of error. Over multiple trials, the network may fall into multiple different minima, with the probability remaining unchanged between trials. This creates the issue that, if it falls into the same minimum several times, the variance of the output error will falsely appear to be zero. The existence of other possible minima cannot be determined until a trial eventually causes one to occur. This makes optimisation difficult since the error from CASPER (which is what we want to minimise with an algorithm) is essentially a random variable with a highly irregular multimodal distribution.

Finally, since each evaluation of the error requires an entire network to be trained, the number of times it can be evaluated is limited by computational resources. Since this is a difficult search space, we consider evolutionary algorithms.

Background

Evolutionary algorithms are a class of algorithm inspired by Darwinian evolution in biology; and they share many features with Darwinian evolution at a conceptual level. In such algorithms, candidate solutions generally represent individuals, these solutions have some form of mutation (random change) and breeding (generating an individual with some combination of the characteristics of the two 'parent' individuals). Additionally, some measure of *fitness* is used, and the least fit individuals are generally discarded to make room for new individuals [7].

In general, evolutionary algorithms are best used for optimisation on difficult search spaces, for example when neither a gradient nor a heuristic are available [7]. For this reason, it was decided that an evolutionary algorithm should be used to optimise the CASPER parameters.

Genetic algorithms are one of the most commonly used types of evolutionary algorithm, and are suited to optimising numerical parameters over a fixed range of values [7], so it was decided that a genetic algorithm should be used for this problem. The algorithm used was custom-designed for the tasks based on the general steps of genetic algorithms.

Initialisation

When determining the parameter ranges, all values within the ranges were considered valid possibilities. As a result, values are initialised using a uniform or log-uniform distribution, as specified in *table 3*.

Mutation

Two types of mutation are used in this algorithm: small and large. In a large mutation, the mutated value is discarded and re-drawn from the range using the same distribution used for initialisation. In a small mutation, the value is redrawn from a small fraction of the possible range. For log-space variables, this range is also taken in log-space.

The purpose of large mutations is to ensure that the full search space is thoroughly explored, even if a large percentage of the population remains close to a single local optimum; however large mutations were found to commonly produce extremely unfit individuals, which were immediately discarded. The probability of this mutation type was thus set to a relatively small values. The purpose of small mutations is to encourage local search and solve the problem of the entire population remaining 'stuck' at a near-optimal value of one variable. Since this much more likely to produce fit

individuals than the large-scale mutation, it was given a higher probability of occurring in all rounds. Favouring large mutations favours exploration, and favouring small mutations favours exploitation.

Culling and breeding

The culling and breeding method used in this algorithm is designed to focus mostly on the best individuals, but allow some degree of randomness to allow suboptimal regions of the search space to be traversed so that new local optima can be found. After each generation, the top 10% of individuals are allowed to survive to the next generation, and the remaining 90% are created by breeding. Keeping more than one individual was considered important due to the stochastic nature of fitness, since the fittest individual in one generation may rank lower in a subsequent generation. Additionally, keeping multiple good individuals will likely improve the quality of the offspring.

Breeding takes place by combining the genetics of randomly chosen pairs of eligible individuals, under the condition that no individual may breed with itself. Individuals are guaranteed to be eligible for breeding if they are above the 60^{th} percentile of fitness. The remaining individuals have a 20% chance of being eligible for breeding. This aims to make most of the individuals relatively fit, but also allow a small number of very unique individuals to exist to maintain diversity.

CASPER hyperparameters

The CASPER hyperparameters considered for optimisation were: N_{hidden}, N_{iter}, k, *L1, L2,* and *L3.* The parameters inherited from RPROP were not optimised as this would likely make the search space too complex and lead to too many interdependencies between the variables. For example, increasing η^+ would decrease the delay before training the *L2*- and *L3*- weighted connections, thereby requiring *L2*- and *L3* to decrease in comparison to *L1* to compensate.

For many of the parameters, the desired range of values to explore spanned many orders of magnitude, and it was not obvious which order of magnitude would be most likely to contain the optimal solution. This would make a uniformly distributed search highly suboptimal, since it would spend nine tenths of its time in the highest order of magnitude, and only one tenth in the next order down, so the lowest orders of magnitude would almost never be explored. Instead, a log-uniform distribution was used in these cases, which was calculated using the formula

$$v_{rand} \sim 10^{\mathrm{U}(\log_{10}(v_{min}),\log_{10}(v_{max}))}$$

Where *U* is the uniform distribution. This ensured that all relevant orders of magnitude would be given approximately equal consideration.

Table 3: CASPER hyperparameters optimised with the genetic algorithm and the ranges considered.

Parameter	Range	Space	Justification
N_{hidden}	[0,5]	Linear	An upper limit was set at length 5, since this is more than the number of neurons preliminary testing showed was necessary to achieve good results. It is linear since all lengths should be given equal consideration.
N_{iter}	[10,100]	Log	Due to the simplicity of the problem and preliminary testing, it was determined that no more than 100 iterations per neuron should be needed. This was confirmed since the optimal number of iterations was found to be significantly fewer than 100.
k	$[10^{-6}, 1]$	Log	The optimal value was not clear from the CASPER paper [1], so a large-range was used, and a log search was used because even the best order of magnitude was unclear.
$L1$	$[10^{-4}, 2]$	Log	Since RPROP can exponentially grow or shrink the step size, these values act as learning delays more than learning rates. As a result, a large range of values should be considered, and log-space is used because the delay is logarithmic with respect to the difference of these values.
$L2$	$[10^{-4}, 10^{-1}]$	Log	
$L3$	$[10^{-4}, 10^{-1}]$	Log	

Combining genetics

With the exception of L1, L2, and L3, the CASPER hyperparameters do not have any natural ordering relative to their function so, for consistency, it was decided that each of them should be handled independently. In contrast, order-dependent combination strategies such as one- or two-point crossover would artificially impose relationships between some of the variables by those nearby in the ordering to be more likely chosen together.

In order to make the combination order-independent, the genomes were represented as a sequence of separate numeric variables rather than a single binary string, and child genomes were generated by choosing each variable randomly from one of the parents with equal probability.

Triple-round execution

Due to the stochastic nature of the fitness function, a trade-off must be made between the accuracy of the evaluation of the fitness function (number of iterations) and the amount of times fitness is evaluated by the evolutionary algorithm. Since the quality of individuals improves over generations, a three-step process is used. Additionally, the hyperparameters define a choice between exploration and exploitation. To balance these factors, a three-round approach was used.

Round 1 focuses more on exploration. It uses a large number of individuals, large mutations, and a large number of iterations. The aim of this round was to eliminate most of the suboptimal regions and to have individuals in most of the optimal regions. Round 2 is more balanced, using more accurate fitness evaluation and moderate mutations. Round 3 focuses on exploitation and evaluation. It uses very accurate evaluation and small mutations.

The values of the hyperparameters used in each round are shown in *table 4*. The varied CASPER hyperparameters and the ranges considered are shown in *table 3*, and the fixed CASPER hyperparameters are shown in *table 2*. Ideally, more genomes and generations would have been used, but there were insufficient computational resources.

Table 4: hyperparameters for the evolutionary algorithm

Parameter	**Value**			**Meaning**
	Round 1	**Round 2**	**Round 3**	
$N_{genomes}$	20	10	10	Number of individuals present in each generation
Keep_frac	10%	10%	10%	Top *% of individuals survive to the next generation
Breed_frac	40%	40%	40%	Top *% of individuals enter the breeding pool
Unfit_breed_prob	20%	20%	20%	*% of remaining individuals also enter the breeding pool
Mutate_prob	10%	3%	0%	Chance of a large mutation occurring
Mutate_nudge_prob	20%	20%	20%	Chance of a small mutation occurring
Mutate_range	10%	3%	1%	Fraction of the range that a small mutation can shift values by.
Sample_count	1	4	16	Number of times to call the fitness function to determine fitness.
$N_{generations}$	30	15	2	Number of generations to use in each round

3 Results and discussion

3.1 Classification

The data were found to be linearly separable, and the network was able to successfully classify all points in both the training and test sets with no hidden neurons. This typically required only a few iterations to produce a correct classification, but the MSE continued to decrease in subsequent iterations until it was almost exactly zero. The MSE convergence was almost twice as fast in terms of iterations on the original fuzzy dataset as it was on the defuzzified dataset.

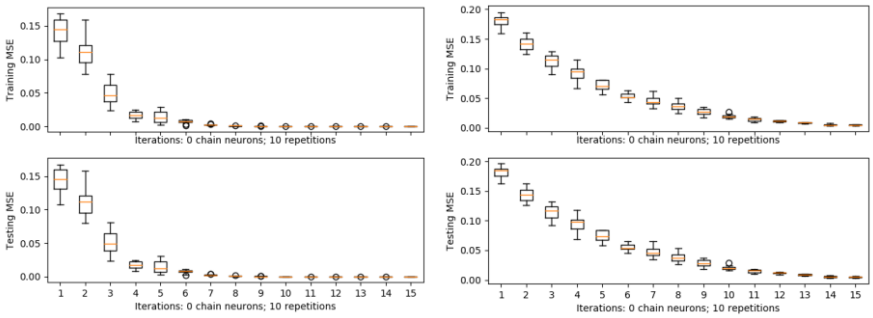

Figure 4: mean squared error plot for the SARS dataset with the original fuzzy labels (left) and the defuzzified labels (right).

The difference in the MSE decay rates can likely be attributed to the presentation of the data. In the defuzzified representation, there are fewer features and the differences between the features are often compressed, so CASPER would require several iterations to increase the step size Δ. Additionally, the fuzzy data provides immediate access to a feature representing whether a value is near the middle of its range for most cases, and this would take multiple hidden neurons to reproduce in the defuzzified data; however this feature is not necessary for the classification.

The authors of the SARS dataset [3] were also able to achieve perfect classification on the dataset, however the methods are not directly comparable.

3.2 Performance of the evolutionary algorithm

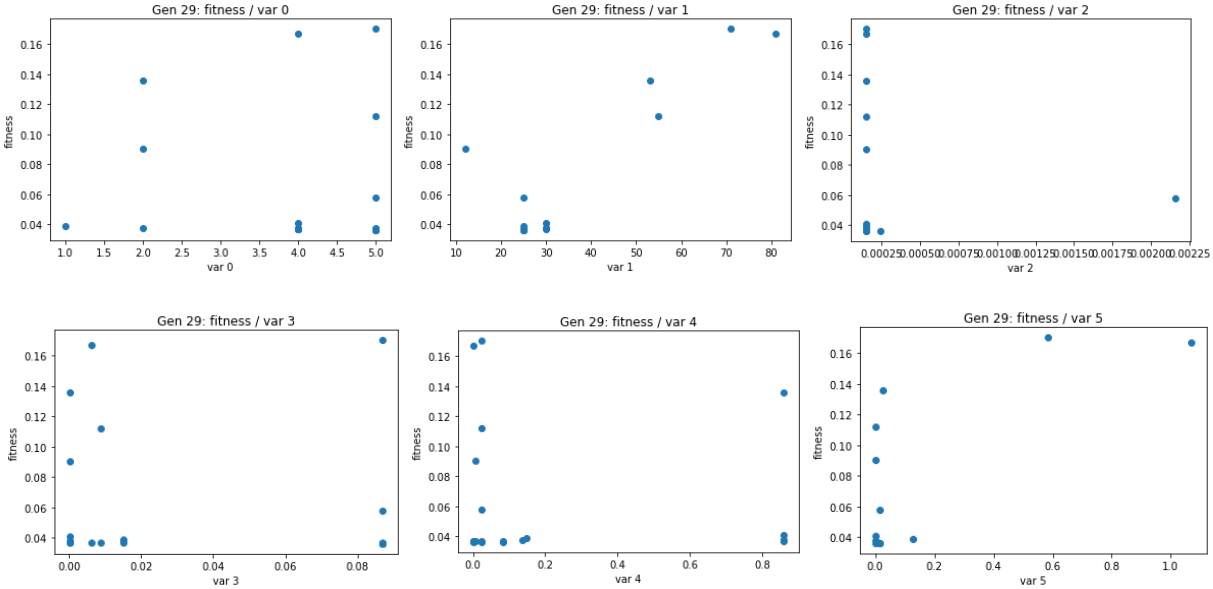

Figure 4: results after round 1 for the six variables: N_{hidden}, N_{iter}, k, L1, L2, *and* L3.

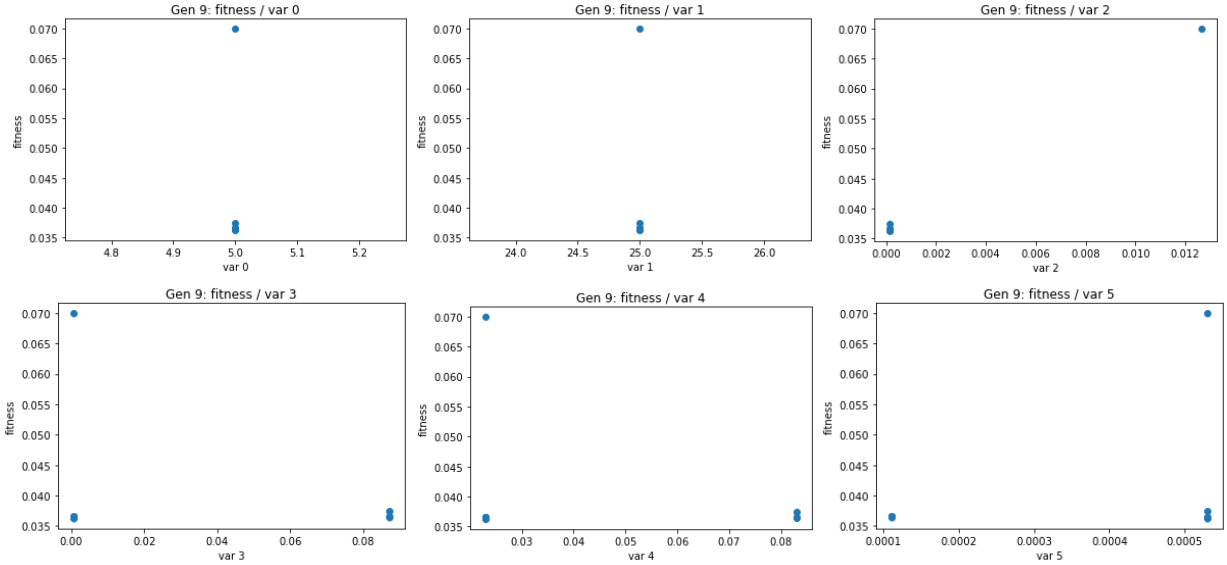

Figure 5: results after round 2 for the six variables: N_{hidden}, N_{iter}, k, L1, L2, *and* L3.

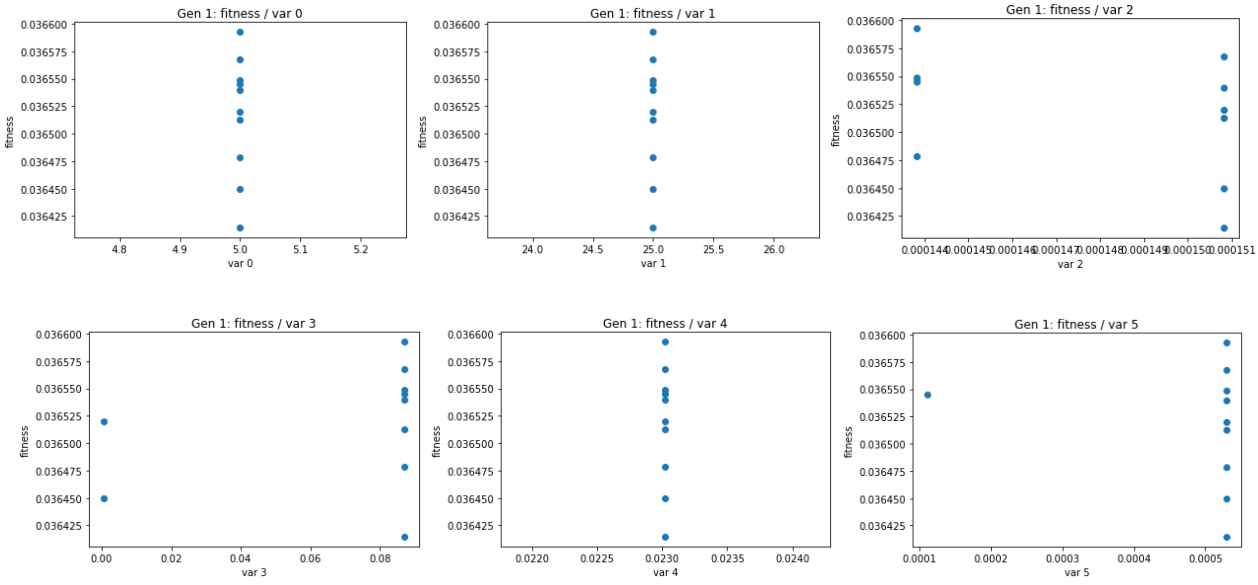

Figure 6: results after round 3 for the six variables: N_{hidden}, N_{iter}, k, L1, L2, *and* L3.

The individuals produced by the evolutionary algorithm at the end of each of the three rounds are shown in *figures 4-6*. Throughout the first round, the population is highly diverse, with the diversity decreasing over the generations as more of the individuals appear to enter local minima. During the second round, the diversity in several of the variables reduces to near a single value, and others approach a bimodal distribution; and this pattern continues through the third round. By the end of the third round, *k* and *L1* have clearly bimodal distributions.

In general, this is how the algorithm was intended to perform, however the lack of small-scale diversity towards the end of the third generation makes it unclear whether the values are truly optimal. This could either be explained by the small-scale mutations being too large and to impossible, or by the first round already having produced near-optimal solutions. Finally, the stochastic nature of the fitness function and the lack of computational resources for more evaluations of it made it difficult to draw robust conclusions about the optimality of the final results, especially given how small the differences between some of the values.

3.3 Identified parameters for CASPER

Table 5: CASPER hyperparameters representing the top 10 genomes after all three rounds were finished.

Rank	N_{hidden}	N_{iter}	k	$L1$	$L2$	$L3$
1	5	25	1.5081×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
2	5	25	1.5081×10^{-4}	6.3869×10^{-4}	2.3024×10^{-2}	5.2906×10^{-4}
3	5	25	1.4383×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
4	5	25	1.5081×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
5	5	25	1.5081×10^{-4}	6.3869×10^{-4}	2.3024×10^{-2}	5.2906×10^{-4}
6	5	25	1.5081×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
7	5	25	1.4383×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	1.1141×10^{-4}
8	5	25	1.4383×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
9	5	25	1.5081×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}
10	5	25	1.4383×10^{-4}	8.6971×10^{-2}	2.3024×10^{-2}	5.2906×10^{-4}

3.4 Reliability of results

Due to the highly stochastic nature of the fitness function, the reliability of the predicted hyperparameters is questionable. As can be seen in *figure 6*, the variation in fitness in the final round is extremely small, and the genomes ranked 1st, 4th, 6th, and 9th are all identical. This suggests that the final round was primarily choosing individuals based on the noise in the fitness function.

The reliability is further called into question because some of the remaining genomes have substantially different parameters. For example the 2nd and 5th placing genomes have an *L1* approximately 1/100 of that of the others, and 7th has an *L3* approximately 1/5 of the others. This is not consistent with the expected optimal values, because CASPER's authors [1] clearly intended for *L1* to be substantially larger than the other two learning rates, not more than an order of magnitude smaller than *L2*. It should be noted, however, that these learning rates are initial step sizes for RPROP, which exponentially grows or shrinks them based on the sign of the gradient [5], thus smaller values effectively function as learning delays.

This discrepancy in the learning rates can be explained by the simplicity of this classification task. Since the different classes are easily separable before adding noise, and the dataset is very large for the number of features available, it is likely that there is a simple classification boundary and a clear gradient towards it. If so, the choice of learning rates would be relatively unimportant as CASPER would likely produce a similar solution anyway, which suggests that the identified learning rates are neither meaningful nor generalisable.

The alternative possibility is that the bimodal values of *L1* are meaningful and represent a useful configuration of CASPER, where the weights of the new neuron are updated after its bias and the new weights to the output neurons. This is unlikely since is conceptually very different to regular CASPER, however it is worth investigating as it may provide new insights about the algorithm.

3.5 Validation

The network with the optimal parameters produced an MSE of about 0.036 on the noisy data, which is very low considering how much noise was added to the data. This suggests that robust classification was achieved. In order to verify this, however, two separate tests were conducted.

First, the CASPER hyperparameters from the optimal genome were used for a CASPER network trained and tested on the original data. This classified all test data correctly and had a very small MSE (typically less than 10^{-10}), which shows that the hyperparameters are valid and generalise to at least a limited extent beyond the noisy data.

Second, a CASPER network trained on the noisy data with these hyperparameters and tested on the original data. This also correctly classified all of the test data, but had a larger MSE of around 0.007. The larger MSE can be explained by the fact that it was trained on extremely noisy data, and therefore will usually make weaker predictions.

Although it is expected that the network trained on the noisy data would generalise better to an extended dataset including other types of medical conditions, this cannot be tested without such a dataset available.

4 Conclusion and future work

4.1 Conclusion

This paper aimed to optimise a CASPER network to robustly classify the patients in the SARS dataset into their medical diagnoses based on their measurements and observations. Due to the simplicity of the dataset, a trivial CASPER network was able to classify it with perfect accuracy, however the simplicity of the dataset meant that such a classifier may not generalise well to noisier, more realistic measurements.

A more robust classifier was developed by adding a substantial amount of noise (Gaussian, standard deviation 0.6) to the data, then using CASPER in combination with an evolutionary algorithm for hyperparameter optimisation to fit it. This robust classifier was successful, as it was able to fit the noisy test data reasonably well (MSE ~ 0.036) and still provide perfect classification on non-noisy test data.

The evolutionary algorithm appeared to be reasonably effective, however it was difficult to be certain due to the stochastic nature of the optimisation task, and the limited computational resources available to execute it. The CASPER hyperparameters resulting from the algorithm were questionable, with the second-place genome containing an extremely small $L1$ value similar to $L3$ and much less than $L2$. This was determined to be most likely either due to lack of accuracy of the fitness function in the evolutionary algorithm or due to the simplicity of the dataset meaning that almost any choice of learning rate is capable of producing good results. If this value is not an aberration, however, it could be a novel and useful configuration of CASPER.

4.2 Future work

First, the classifier produced in this paper was very accurate and robust on the dataset, however the dataset itself was very easy to classify. The Gaussian noise / CASPER / evolutionary optimisation methodology used to produce the robust classification should also be applicable to other datasets, however it is unclear how effective it will be when the original data are more difficult so separate, such as when multiple medical conditions present with similar measured symptoms. As a result, future work is recommended in exploring the applicability of these methods to an extended dataset containing more medical conditions. This work would most likely need to make adjustments for unbalanced classes, for example, taking more repetitions of the data in the smaller classes (before adding Gaussian noise).

Second, the evolutionary process appeared to be capable of producing noteworthy results about CASPER's hyperparameters, however it is unclear how well these results would generalise to other datasets and whether the SARS dataset required any features complex enough to warrant such optimisation. To produce more useful result and test the unlikely hypothesis that $L1 \sim L3 \ll L2$ is a useful configuration for CASPER, the same method could be applied using a many different datasets from unrelated classification tasks, with the fitness function modified to determine the mean error across all of them. Then, the hyperparameters identified by the evolutionary algorithm would be likely to generalise well to other tasks. Validation could be conducted by testing the hyperparameters on a new classification dataset not used during training.

References

1. Treagold, N.K., Gedeon, T.D.: A Cascade Network Algorithm Employing Progressive RPROP. UNSW School of Computer Science and Engineering (n.d.)
2. Prechelt, L.: Investigation of the CasCor Family of Learning Algorithms. Universitaet Karlsruhe (1996)
3. Mendis, B. Gedeon, T., Koczy, L.: Investigation of Aggregation in Fuzzy Signatures. ANU Department of Computer Science and BUTE Deparement of Telecommunication and Telematics (n.d.). Proceedings, 3rd International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore. (2005)
4. Wong, K., Gedeon, T., Koczy, L.: Construction of Fuzzy Signature Data: An Example of SARS Pre-clinigal Diagnosis System. NTU, BUTE, ANU, IITEE (n.d.)
5. Braun, H., Riedmiller, M.: RPROP – A Fast Adaptive Learning Algorithm. Universitaet Karlsuhe (n.d.)
6. Zadeth, L.: Fuzzy Sets. University of California. Information and Control 8, 338-353 (1965)
7. Vikhar, P. Evolutionary Algorithms: A Critical Review and its Future Prospects. Shri Sant Gadge Baba Amravati University. International Conference on Global Trends in Signal Processing, Information Computing and Communication. (2016)