CS7641 Assignment 2 - Eric Gregori (03/11/18)

Optimization Problems

Optimization problems data is gathered using the ABAGAIL library [1]. A custom training class (emgEqualityTrainer) is to measure the number of algorithm evaluations required to find the optima. Each algorithm is instantiated 'run' times. With each instantiation choosing new random values specific to the algorithm.

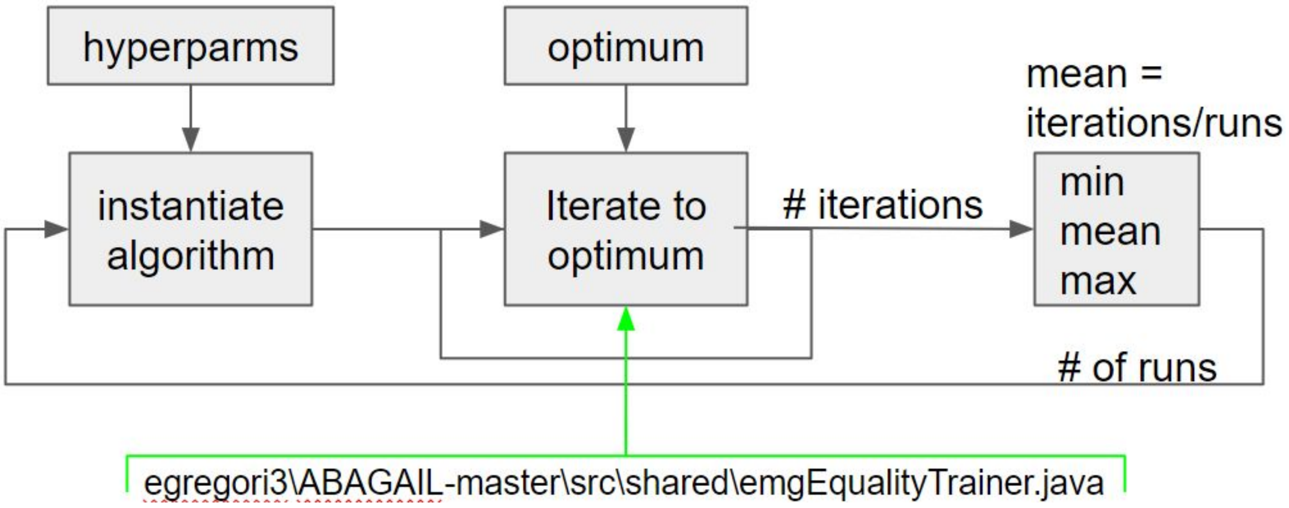


Figure 1. Optimization problem data collection (mean # of fitness score evaluations after 1000 runs)

For optimization problems, the term 'trainer' is misleading. emgEqualityTrainer iterates the algorithm until the algorithm returns the optimum. This requires the optimum to be know in advance. For each iteration, emgEqualityTrainer calls the algorithms train() function and its getOptimal() function with the goal of measuring the number of fitness score evaluations the algorithm required to find the optima.

The two color problem is interesting because it is easy to visualize. The goal is simply to create an alternating pattern of two values. This should be the simplest problem for the algorithms to solve yielding a nice baseline of comparison.

Four peaks is an interesting problem because it has very narrow global optima peaks. The majority of positive slopes lead to suboptimal local maxima. With such a small target, random restart algorithms have a significant disadvantage to landing on a positive slope leading to a global optima.

"*The n-Queens problem is to place n chess queens on an n by n chessboard so that no two queens are on the same row, column or diagonal.*" [3] The N-queens problem has been a controversial problem in AI over whether it is a NP- and #P-Complete problem. [3]

RHC	SA (rate=0.95)	GA (P1, P2, P3)	MIMIC (P1, P2)

No parameters	rate = multiplier for temperature decrease per iteration.	P1 = population size
		P2 = tomate / iteration
		P3 = mutate / iteration
			P1 = samples / iteration
			P2 = samples to keep

Table 1. Algorithm Parameters

Two Colors - egregori3\ABAGAIL-master\src\opt\example\TwoColorsEvaluationFunction.java

The two colors problem has two global maxima out of 2^N possibilities. **The global maxima peaks are wide, providing ample area for the stochastic algorithms to randomly land on**. The large amount of positive slopes leading to a global maxima means that most of these algorithms should perform well.

optimal(P) = N - K

Success = found global maxima

1000 runs	**RHC**	**SA (rate=0.95)**	**GA (N, N/2, N/10)**	**MIMIC (N, N/10)**

K = 2
N = 20
P = 18
	1000 successes
	61 evaluations
		1000 successes
		107 evaluations
			709 successes
			10 evaluations
				1000 successes
				9 evaluations

K = 2
N = 40
P = 38
	1000 successes
	162 evaluations
		1000 successes
		220 evaluations
			336 successes
			38 evaluations
				1000 successes
				23 evaluations

K = 2
N = 80
P = 78
	997 successes
	404 evaluations
		995 successes
		470 evaluations
			59 successes
			108 evaluations
				1000 successes
				36 evaluations

K = 2
N = 160
P = 158
	585 successes
	788 evaluations
		158 successes
		828 evaluations
			0 successes	1000 successes
				84 evaluations

time	minutes	minutes	minutes	hours

Table 2. 1000 runs (each run is a new instantiation of the algorithm) (mean # of fitness evaluations)

RHC	The two color problem has wide global maxima peaks so there is a higher probability it will randomly land on a slope leading to a global maxima. This explains why RHC did so well on this problem.
SA	As with RHC, SA did very well on this problem. Since SA is RHC with timed random restart, It is expected that if RHC does well on a problem SA should as well.
GA	Single crossover GA performed very poorly on this problem. Failing to find any optima when the other three algorithms did. This may be do to parameter selection. With N=160, the population size is 160, tomate/iteration is 80, and the mutate/iteration is 16.
MIMIC	This problem is far to simple for MIMIC. Although MIMIC is very successful at solving this problem, it is not very efficient in terms of compute cycles/time.

Table 3. Two Colors algorithm analysis

Two Colors Conclusion

The two color problem was chosen specifically because it is relatively easy. The goal is to highlight the strength of RHC, which it did. The two color problem is an example of the type of problem RHC excels at. Problems with wide global optima peaks resulting in many potential landing spots for the stochastic dependency of RHC.

GA's failure on such a simple problem is interesting.

Four Peaks - egregori3\ABAGAIL-master\src\opt\example\FourPeaksEvaluationFunction.java

"*There are two global maxima for this function. They are achieved either when there are T + 1 leading I's followed by all O's or when there are T + 1 trailing O's preceded by all 1 'so There are also two suboptimal local maxima that occur with a string of all I's or all O's.* " [2] **Skinny peaks surrounding its global maxima.**

optimal(P) = N + (N-T-1)

Success = found global maxima

1000 runs	**RHC**	**SA (rate=0.95)**	**GA (N, N/2, N/10)**	**MIMIC (N, N/10)**

N = 10
T = 1
P = 18
	577 successes
	55 evaluations
		914 successes
		412 evaluations
			999 successes
			112 evaluations
				1000 successes
				23 evaluations

N = 20
T = 2
P = 37
	331 successes
	272 evaluations
		858 successes
		763 evaluations
			623 successes
			524 evaluations
				993 successes
				123 evaluations

N = 40
T = 4
P = 75
	17 successes
	758 evaluations
		3 successes
		915 evaluations
			11 successes
			800 evaluations
				931 successes
				366 evaluations

N = 80
T = 8
P = 151
	0 successes	0 successes	0 successes	382 successes
				768 evaluations

time	minutes	minutes	minutes	hours

Table 4. 1000 runs (each run is a new instantiation of the algorithm) (mean # of fitness evaluations)

RHC	Random Hill Climbing does fairly well on four peaks if executed enough times. This makes sense. If you randomly drop a climber on a mountain range enough times, eventually they will land near the highest peak.
SA	Simulated Annealing finds the global maxima twice as often as RHC. This is expected because SA is initially able to jump out of local maxima. Making it less dependent on the initial random position.
GA	Single crossover Genetic Algorithm performs closer to SA than expected.
MIMIC	MIMIC consistently found the global optimum in the least number of evaluations. MIMIC stood out as the problem complexity increased finding the global maxima 93.1% of the time within 1000 evaluations when N=40.

Table 5. Four peaks algorithm analysis

Four Peaks Conclusion

Four peaks has 2 global maxima out of 2^N possibilities. The four peaks problem has skinny peaks surrounding its global maxima. These skinny peaks provide little positive slope for the stochastic algorithms to randomly find. As expected, MIMIC stood out with this problem. Finding the global maxima in every case tested and in significantly less evaluations than the other algorithms. Also, as expected, MIMIC took significantly longer to solve the problems due to the number of calculations required by the algorithm.
Using the analogy of a person randomly parachuting onto a mountain range. The skinny peaks provide very little area relative to the rest of the mountain range for the parachuter to land. The jumper is much more likely to land on a slope leading to a local maxima as opposed to a global maxima. MIMIC's ability to statistically "map" the terrain provides the algorithm guidance for where to "aim" the jumper.

NQueens - egregori3\ABAGAIL-master\src\opt\ga\NQueensFitnessFunction.java [4]

The NQueens problem has multiple solutions (global optima). The fitness function counts the number of non-attacking pairs of queens. [1] An equation for the optima could not be found. A optima table was found here: [4]

Success = found global maxima

1000 runs	**RHC**	**SA (rate=0.1)**	**SC GA (200,0,10)**	**MIMIC (200,10)**

N = 10
P = 45
	273 successes
	83 evaluations
		594 successes
		162 evaluations
			995 successes
			179 evaluations
				16 successes
				38 evaluations

N = 20
P = 190
	242 successes
	366 evaluations
		444 successes
		333 evaluations
			5 successes
			639 evaluations
				0 successes

N = 30
P = 435
	179 successes
	631 evaluations
		265 successes
		590 evaluations
			0 successes	0 successes

N = 40
P = 778
	779 successes
	512 evaluations
		849 successes
		480 evaluations
			0 successes	0 successes

time	minutes	minutes	hours	hours

Table 6. 1000 runs (each run is a new instantiation of the algorithm) (mean # of fitness evaluations)

Success = found global maxima

1000 runs	**SA (rate=0.05)**	**SA (rate=0.5)**	**SA (rate=0.95)**	**SA (rate=0.99)**

N = 30
P = 435
	267 successes
	557 evaluations
		424 successes
		618 evaluations
			424 successes
			682 evaluations
				283 successes
				761 evaluations

N = 40
P = 778
	859 successes
	473 evaluations
		910 successes
		469 evaluations
			927 successes
			498 evaluations
				805 successes
				685 evaluations

Table 7. SA Tuning

Success = found global maxima

crossover	**A,A/2,A/10**	**A,A/10,A/2**	**A,A/10,A/10**	**A,A/20,A/20**	**A,0,A/2**	**A,0,A/5**	**A,0,A/10**	**A,0,A/20**
single	🟥	🟥	🟥	🟥	1.0.7.5	0,1,0,5	0,0,7,4	0,0,7,11
uniform	🟥	🟥	🟥	🟥	0,0,2,6	1,0,5,4	0,1,5,10	0,1,8,9
2 point	🟥	🟥	🟥	🟥	0,2,1,3	0,1,4,7	0,1,1,7	0,2,11,14

Table 8. GA Tuning N=20, P=190 (A=50,100,200,400) (successes)

Success, 45 iterations

	Samples to keep					
Samp/iter	**1**	**A/50**	**A/25**	**A/10**	**A/5**	**A/2**
50	🟥	🟥	🟥	🟥	🟥	🟥
100	🟥	🟥	🟥	🟥	🟥	🟥
300	🟥	🟥	12,45	🟥	🟥	🟥
400	🟥	15,56	37,89	62,49	17,78	19,95

Table 9. MIMIC Tuning N=20, P=190 (A=50,100,300,400) (successes)

RHC	RHC did a good job. Which makes sense because the problem has multiple solutions (global optima) providing a higher probability that the algorithm would randomly find a slope leading to the global peak..
SA	This problem highlights the advantages of SA. As shown in table 7, if the temperature schedule is too aggressive or not aggressive enough SA acts like RHC. The correct temperature schedule can increase the performance of SA significantly.
GA	As shown in table 8, GA was tested with a range of parameters. Based on the data, GA is only able to find a global optima if the toMate/iteration parameter is set to 0. This provides further evidence that the problem landscape has a lot of flat surfaces. The distributed nature of GA would cause a dilution of information when mating. A better strategy would be random mutation, a theory backed up by the data in table 8.
MIMIC	MIMIC took many days to evaluate on the NQueens problem. Each test took hours.

Table 10. NQueens algorithm analysis

NQueens Conclusion

Based on the data, I think the NQueens problem has a lot of flat surfaces (points of equal fitness). This would explain why GA and MIMIC has such a hard time with the problem. Flat surfaces occur when neighboring points have the same fitnes value. These flat surfaces provide no data to the algorithm. For these problems, the more random the optimizer, the better it's performance.
Flat surfaces would also explain why GA does better when mating is disabled. Multiple "cells" on a flat surface would dilute the information from "cells" on or near peaks when mating. Mutation provides the random "exploring" in GA. This random exploring is more valuable for a problem with lots of "flat" space.

Optimization Problems Summary

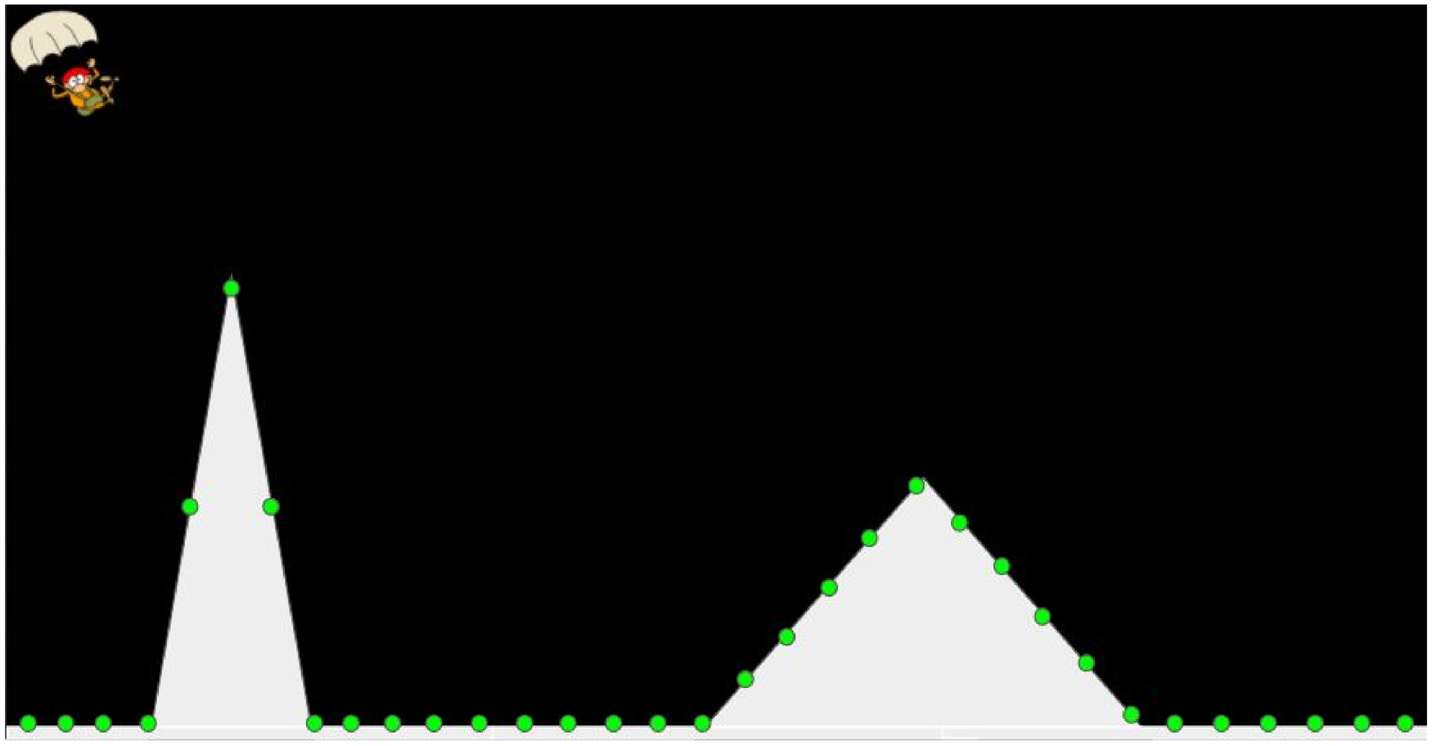


Figure 2. Visualizing stochastic optimizers

	RHC	SA	GA	MIMIC
two-colors wide-optima	wide optima increase effectiveness	effective but requires more cycles than RHC	not a good choice for such a simple prob	Overkill for such a simple problem
four-peaks thin optima	thin optima peaks reduce effectiveness	no more effective than RHC	inefficient	Very efficient for slim peaks
NQueens flat	multiple optima increase effectiveness	multiple optima increase effectiveness	ineffective	ineffective

Continuing with the parachuter analogy (figure 2). RHC and SA perform better on problems that have wide global optima peaks presenting a large number of landing points on slopes that lead to the global maxima. The two-color problem is an example of a “wide peak global maxima” problem (right peak in figure 2). The large number of landing points on the slope increase the probability of the algorithm landing on a path to the peak.SA and RHC did well on the Two-color problem.
The four peaks problem is a good example of a “thin peak global maxima” problem (left peak in figure 2). Strictly stochastic algorithms do not work well on these problems.

Overall, the best algorithm (based on these three problems) is SA. It works well on wide peaks, flats, and better than RHC on thin peaks. It is significantly faster than MIMIC or GA (with these toy fitness functions). As added benefit, it is easy to implement.

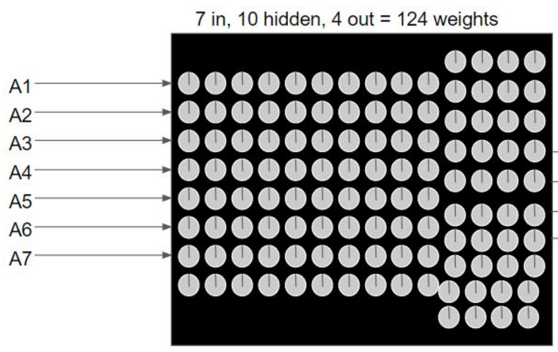
Using SGD, RHC, SA, GA for Neural Network Training

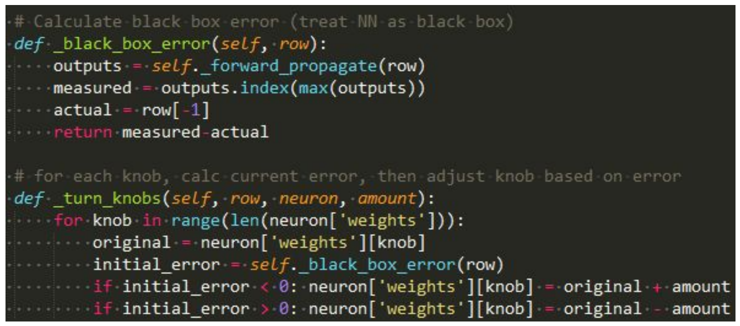
Dataset from assignment #1 - WiFi Localization

	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Class
Min	-74	-74	-73	-77	-89	-97	-98	1
Max	-10	-45	-40	-11	-36	-61	-63	4
Mean	-52.33	-55.62	-54.96	-53.57	-62.64	-80.99	-81.73	

Assignment 1 Neural Network Hyperparameters

layers	hidden nodes	hidden activation	learning rate	learning rate init	max # of epochs	input nodes	input activation	output nodes	output activation
3	10	relu	constant	0.01	200	6	relu	4	softmax

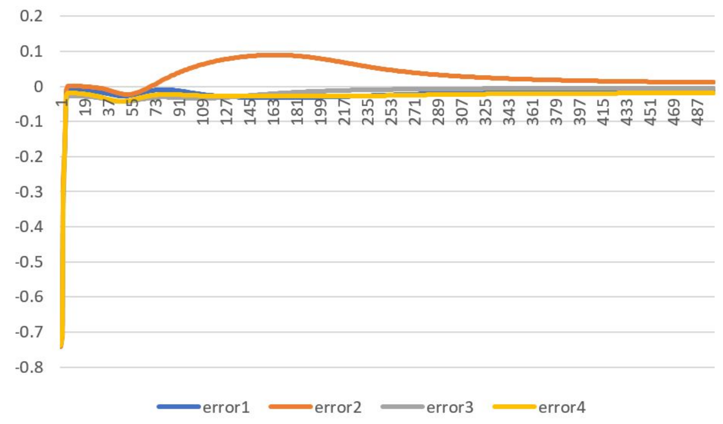




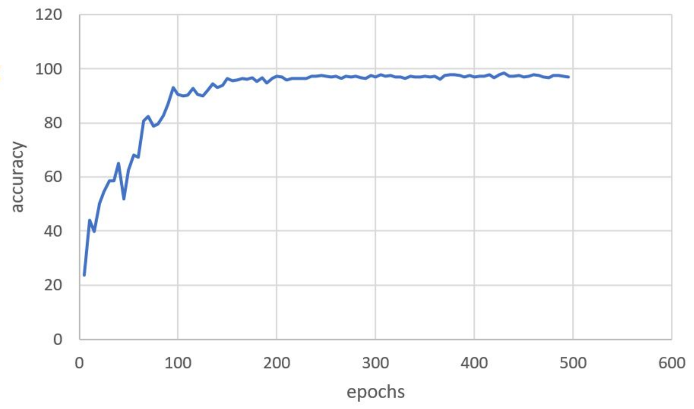
```
# Calculate black box error (treat NN as black box)
def _black_box_error(self, row):
    outputs = self._forward_propagate(row)
    measured = outputs.index(max(outputs))
    actual = row[-1]
    return measured-actual

# for each knob, calc current error, then adjust knob based on error
def _turn_knobs(self, row, neuron, amount):
    for knob in range(len(neuron['weights'])):
        original = neuron['weights'][knob]
        initial_error = self._black_box_error(row)
        if initial_error < 0: neuron['weights'][knob] = original + amount
        if initial_error > 0: neuron['weights'][knob] = original - amount
```

Figure 3. The Neural Network based on the above hyperparameters has 124 knobs (weights)

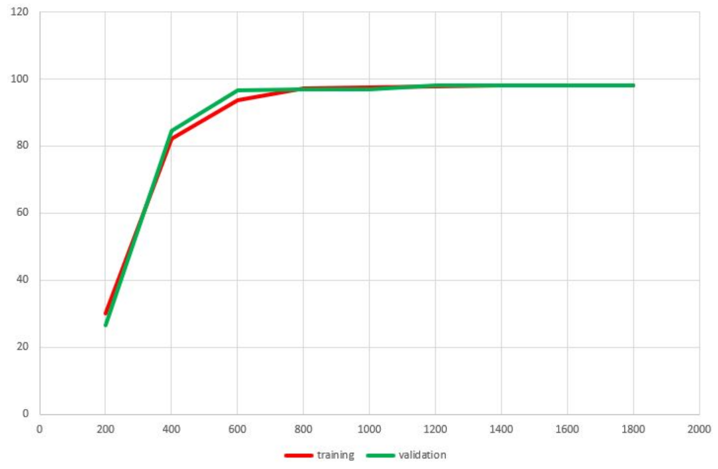
BackPropagation with Stochastic Gradient Descent (SGD)



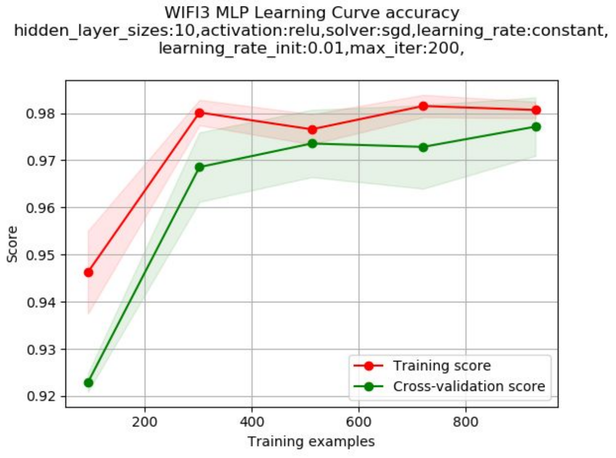
training error/epoch



accuracy/epochs



assignment 2 NN



SciKit-Learn NN

For assignment 1, a SciKit-Learn neural network (MLPClassifier) was trained on the wifi data using the above hyperparameters. The resulting learning curve is shown above. A custom neural network written entirely in Python was developed for assignment 2 (emg/NeuralNet.py). This custom NN and test framework (emg/EvaluateClassifier.py) makes it easier to instrument the training process for analysis purposes. The training and accuracy versus epochs curves provide details on the learning process. [5]

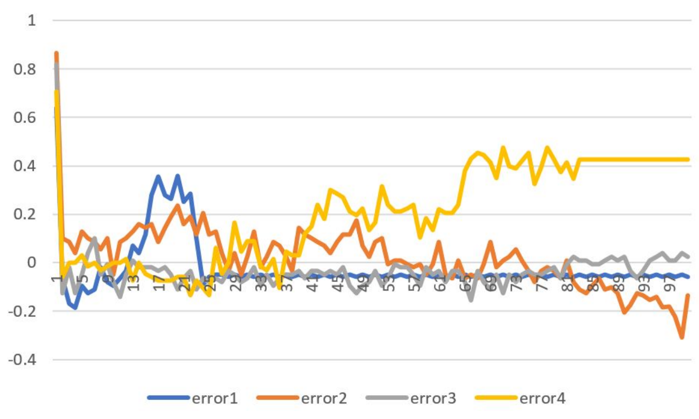
Training error per epoch is a plot of the average training per epoch. During the training process, the error is calculated as the difference between the desired and actual neural net output. This error is accumulated, averaged, and plotted.

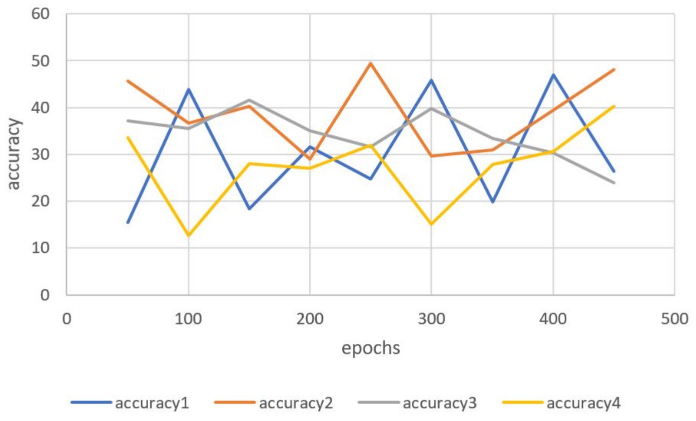
The backpropagation algorithm propagates the error from the network output to it's input, using the error to calculate intermediates values between the network layers. This is possible due to the first derivative and chain rule. The hill climbing algorithms cannot do this. The hill climbing algorithms must treat the neural network as a black box.

The neural network can be configured to either initialize all weights to random values when instantiated, or the weights can be read from a json file at initialization. These feature provides a consistent base for comparing algorithms since the weights all start at the same value.

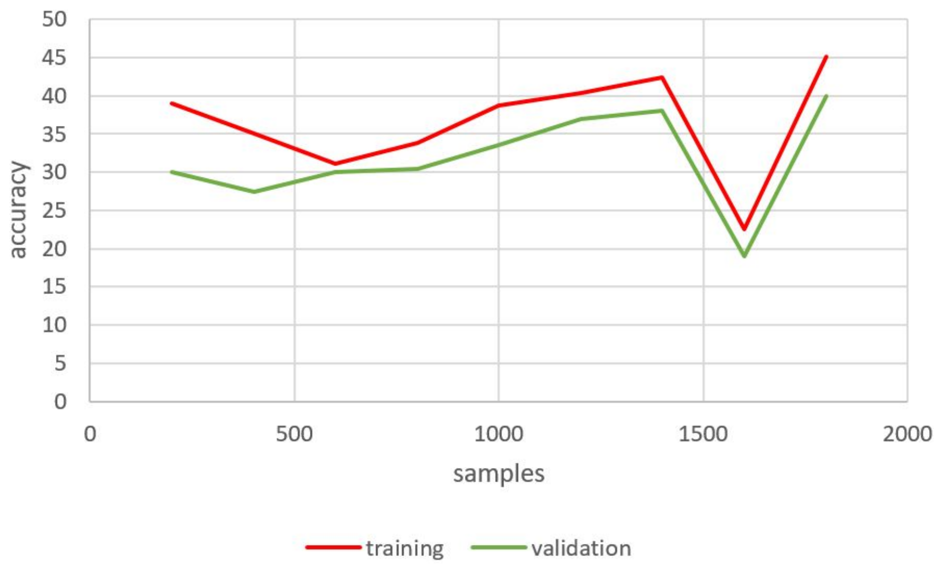
Randomized Hill Climbing

As shown in figure 3, the neural network is a black box with 124 knobs. Each knob varies a weight. The RHC algorithm initially sets the knobs to random values. It applies an input to the black box and subtracts the black box output from the actual output resulting in an error measurement. After each knob is turned, a new error is calculated, and the next knob is turned based on this new error.



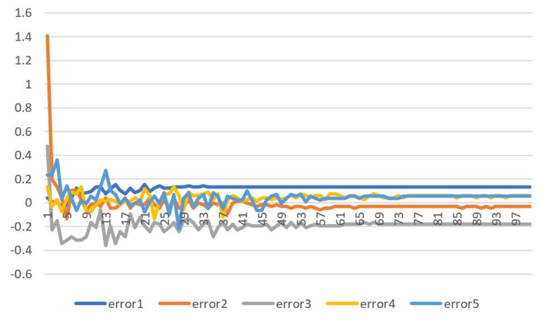


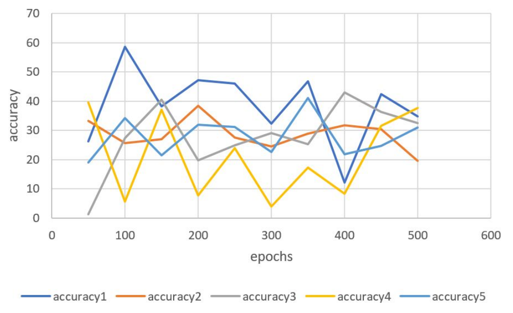
training error/epoch **accuracy/epochs**



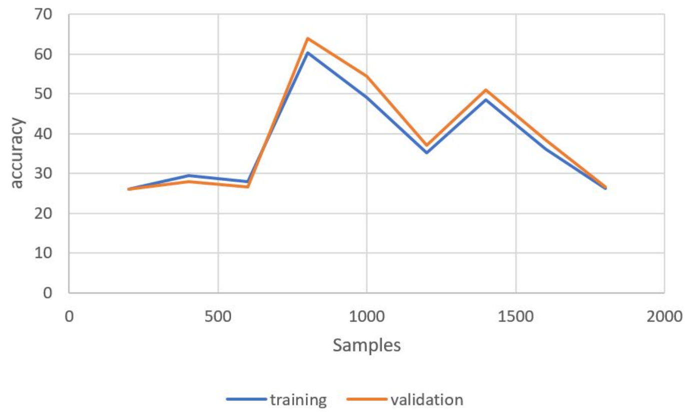
Simulated Annealing

The simulated annealing algorithm builds on the RHC algorithm by adding randomness beyond the initial distribution. [9] The annealing process describe how the additional randomness is applied. The algorithms starts out hot (very random) and cools down (less random). This cooling schedule is obvious in the error curves. The errors start fluctuating wildly and slowly stababilzes.

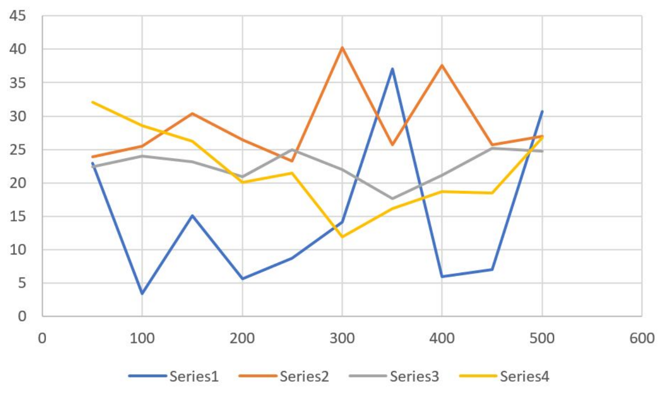




training error/epoch **accuracy/epochs (rate=0.9)**



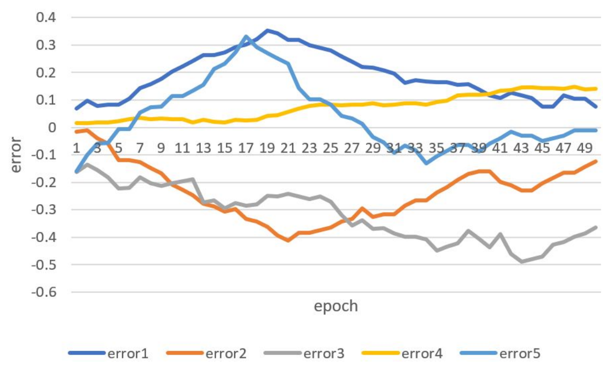
learning curve



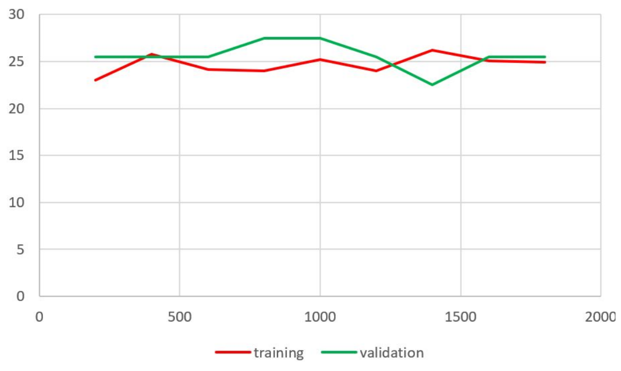
accuracy/epochs (rate=0.5)

Genetic Algorithm

“Survival of the fittest” [8] Create a population of hypothesis, evaluate each hypothesis, the bad ones die, the good ones survive and get to mate. During mating, a portion of the population mutates. GA is implemented in emg/NeuralNet.py, specifically by train_network_ga(). First, a population of hypothesis is created by instantiating multiple neural networks (each NN is a hypothesis), with random weights. Each NN is then evaluated. Very poor performers are pruned. The rest are mated using a crossover function. **The crossover function combines networks by randomly taking neurons from two networks to create a new network.** The new network replaces the pruned networks keeping the population size the same. Finally, a percentage of the weights in the network and randomly changed resulting in a mutation.



training error/epoch



“learning” curve

NN Training Summary

Trainer	best accuracy	epochs for best accuracy	fixed epoch accuracy 150 epochs	training time	description
SGD (SciKit)	98.2%	200	97.9%	minutes	Data from assignment 1
SGD	98.2%	485	97.5%	minutes	Reproduce assignment 1 using new NN
RHC	49.5%	250	40.2%	minutes	Gets stuck in local maxima resulting in inconsistent results.
SA (rate=0.9)	58.6%	100	50.5%	minutes	An improvement over RHC
SA (rate=0.5)	40%	300	30%	minutes	Worse than RHC
GA	25.5%	100	25%	hours	Very difficult to get tuned correctly

Based on the data from RHC and SA, the neural network function has lots of local (suboptimal) maxima. The difference between optima found when SA rate is 0.9 (slow cooling) and SA rate is 0.5 (fast cooling) support the many local (suboptimal) maxima hypothesis. Slow cooling spends more time exploring before settling on a final optima.

GA is an interesting approach to “training” a neural network. The GA approach is to create many neural networks and instead of training them filter them out via evolution to find the neural networks that give the best answers. The hyperparameter tuning and training time has made it difficult to get right.

GA performance was very bad. Based on the learning curve, the neural network did not learn at all. In order for GA to work well, it needs thousands of hypothesis. Evaluating all the hypothesis takes a lot of time. The GA algorithm is a very logic algorithm for parallelization. GA is also sensitive to its hyperparameter tuning. Slight changes in the hyperparameters can have very different results (see training error/epoch plot)..

A better approach than plane GA may be to combine GA and SA. Create multiple hypothesis, train each one with SA, then filter the trained networks using evolution. I did not have time to attempt this approach, but the theory seems solid. Using the parachuter analogy, drop thousand of parachuters on the problem with each one landing and trying to get to the top of the nearest peak.

No approach came close to SGD which directly calculates the appropriate weights, no guessing required.

References

AGAGAIL
1. https://github.com/pushkar/ABAGAIL
Four Peaks
2. http://papers.nips.cc/paper/1328-mimic-finding-optima-by-estimating-probability-densities.pdf
N-Queens
3. https://jair.org/media/5512/live-5512-10126-jair.pdf
4. https://pdfs.semanticscholar.org/9112/c9fa50ee462d08181784e9b9cdf198098dbd.pdf
Neural Networks
5. https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/
Algorithms
7. https://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf
8. https://blog.sicara.com/getting-started-genetic-algorithms-python-tutorial-81ffa1dd72f9
9. http://aima.cs.berkeley.edu/python/search.html
10. https://github.com/aimacode/aima-python
Parachute Cartoon
14. http://laoblogger.com/cartoon-clipart-parachute.html#gal_post_29593_cartoon-clipart-parachute-5.jpg