Project 3 Design Document:

Group 15 Jake Danczyk, Matthew Cirillo, Justin Braun, Henry King

Problem Description

Our task for this assignment is to apply three different evolutionary approaches of training a feedforward neural network and then compare the results to backpropagation. The three algorithms we will be applying are a (+) evolution strategy, differential evolution, and genetic algorithm with real-valued chromosomes. Each algorithm will be trained to solve the following problems:

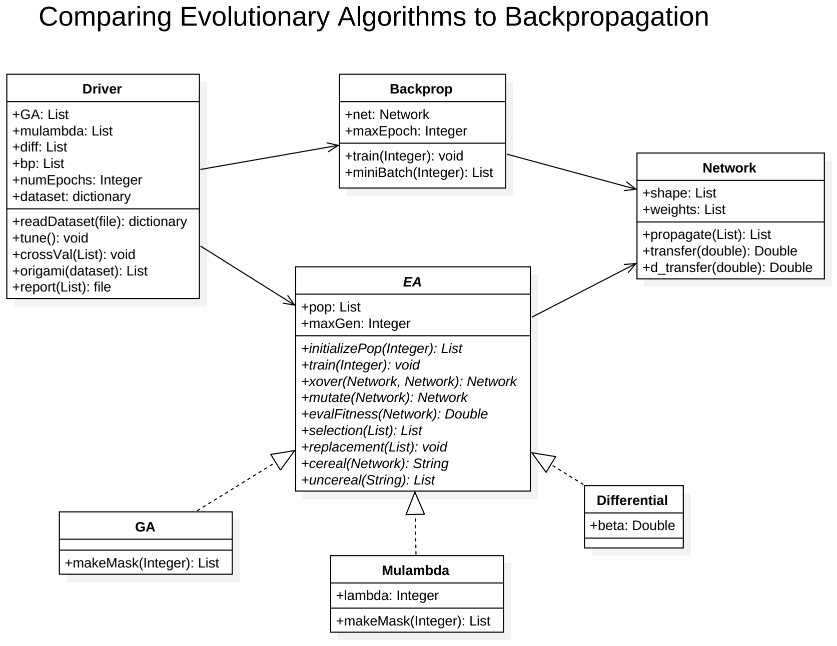
1. The Leaf classification problem. This dataset consists of a collection of shape and texture features extracted from digital images of leaf specimens originating from a total of 40 different plant species. There are 16 attributes in total that each algorithm will learn to identify each leaf.
2. The Wine classification problem. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
3. The Poker Hand classification problem. Each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes.
4. The Zoo classification problem. This data presents 17 boolean valued attributes to describe a category of animal. There are 7 possible categories.
5. The Glass identification problem. Given 9 real valued attributes our algorithm will predict one of 7 possible types of glass.

Hypotheses

In order to evaluate the head to head performance of our algorithms, we will use two evaluation metrics. Performance will be evaluated by percent correct, and convergence rate will be evaluated as the number of iterations until the minimum error was achieved.

We expect the evolutionary algorithms to generally outperform the backpropagation, as evolutionary methods are more likely to avoid being trapped in local minima/maxima. We expect to see generally lower convergence rates for the three evolutionary algorithms, as their already costly training is generally even slower for classification problems (Ježowicz et al., 2015). We predict the genetic algorithm's convergence rate to show improvement relative to the other evolutionary algorithms for the Poker Hand problem, as the discrete variables should result in a lower cardinality alphabet despite the real-valued implementation. We expect the Leaf and Wine datasets to be the most difficult problems, with a number of local optima, and we predict that differential evolution will be the top performer for these problems. This is because differential evolution has improved diversity and capability at escaping local optima compared with + and GA algorithms.

UML



Our design incorporates three major classes and four minor classes. The Backprop class contains the behavior for training a multilayer perceptron with backpropagation. The net object will be the fixed structure network of the Network class and maxEpoch is the maximum number of epochs during training. The train() method will run backpropagation stochastically on batches made by the minibatch() method.

EA is an abstract class that subdivides into the subclasses GA, Mulambda and Differential. EA contains methods for the shared behavior between the three evolutionary algorithms, and the subclasses override these methods with behavior specific to themselves. initializePop() is the method for generating a population and randomizing the initial weights. xover() is the crossover operator and mutate() is the mutation operator. GA and Mulambda will overwrite this to incorporate a binary mask method during crossover. Differential will overwrite this as well, using the trial vector and beta parameter. evalFitness() measures the fitness of the individuals by the loss function. selection() is the method for choosing parents for crossover and replacement() is the method of inserting individuals back into the population. cereal() is the method of representing the adjacency matrices as a serialized string for crossover and mutation and uncereal() is the method for doing the reverse. These serialized matrices are stored in the pop object.

The Driver class contains various tuning and evaluation methods. The driver holds an object for each of the four algorithms to store the current structure and tuning parameters; bp, ga, mulambda, diff. These are adjusted during the tuning process and tune() provides us with a way to quickly set and evaluate the change in parameters. With the finalized shape and parameters, crossVal() is used to perform the head to head comparisons. origami() is our method for folding the dataset during cross-validation. report() collects and writes the average percent correct and the convergence rates to a file for plotting and analysis.

Experimental Approach

The first step will be to determine the overall structure of the neural networks to use for each of the five classification problems. The number of inputs and outputs will determined by the data set being used. We will determine how many hidden layers, 1 or 2, and the number of nodes per layer by trying variants testing the training methods on a small number of batches, of smaller sizes to get a feel for how each change affects the network. We will be specifically looking at the rate at which error is dropping and the percent correct achieved on the batches.

Then, the same procedure will be done for the tunable parameters. Backpropagation has the tunable learning rate and momentum parameters. The evolutionary algorithms have an adjustable population size. (+) will need a to be determined, we will start from the standard : ratio of 1:7 (Kruse, 2016). Differential will need an established beta parameter.

In order to properly measure differences in training algorithms, we are holding constant as much as we can between the three evolutionary algorithms. That means that crossover in GA and (+) will be global, uniform crossover. The probability of mutation will be the same. We'll also be holding population size the same across all three, but acknowledge this could affect the performance on algorithms like Differential that can do better with larger population sizes.

With the structure and parameters in place, we will use 5x2 cross validation to compare the algorithms head to head. Our end results will be gauged by comparing the average iterations until convergence for each method and its final average percent correct. An iteration, we've decided, is the occurrence of a weight update in our networks. For the evolutionary algorithms this would be after the mutation and crossover steps occur. For the backpropagation algorithm, this will be after one instance of backpropagation. To make iterations comparable between the evolutionary algorithms and backpropagation we have decided that the backpropagation batch size be the size of the training dataset, meaning our backpropagation algorithm will be using the "offline" method. This way, for each kind of algorithm, one iteration of the training data (an epoch) corresponds to one optimization of network weights.

The best algorithm will be one with the highest percent correct and with the lowest number of iterations to convergence. This will be determined for each dataset.

References

1. Kruse R., Borgelt C., Braune C., Mostaghim S., Steinbrecher M. (2016) In: Computational Intelligence. Texts in Computer Science. Springer, London
2. Tobias Blickle and Lothar Thiele
   A Comparison of Selection Schemes used in Genetic Algorithms
   TIK-Report Nr 11, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, 1995
3. Michael Herdy.
   The number of offspring as strategy parameter in hierarchically organized evolution strategies. *ACM Sigbio Newsletter*, 13(1): 2-9 , 1993
4. Max Shpak.
   Evolution of variance in offspring number: The effects of population size and migration.
   Theory in Biosciences, 124(1):65-85, 2005
5. Tomáš Ježowicz, Petr Buček, Jan Platoš, Václav Snášel.
   Evolutionary Algorithms for Fast Parallel Classification.
   In *Proceedings of the 9th International Conference on Computer Recognition Systems*, pages 659-670, Wroclaw, Poland, 2015
6. Silva P.F.B., Marçal A.R.S., da Silva R.M.A. (2013) Evaluation of Features for Leaf Discrimination. In: Kamel M., Campilho A. (eds) Image Analysis and Recognition. ICIAR 2013. Lecture Notes in Computer Science, vol 7950. Springer, Berlin, Heidelberg