

Diagnosing Breast Tumor Malignancy with a Genetic Algorithm and RBF Network

Contest entry for “Forecasting & Futurism 3rd Annual iPad Contest”, by Jeff Heaton

Jeff Heaton

636-xxx-xxxx

Reinsurance Group of America

1370 Timberlake Manner Parkway

Chesterfield, MO 63017

Description of Problem

My contest entry makes uses a Genetic Algorithm to devise a Radial Basis Function (RBF) Network to predict the malignancy of breast tumors. This determination is made using the following nine attributes collected from a tumor.

- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
- Marginal Adhesion
- Single Epithelial Cell Size
- Bare Nuclei
- Bland Chromatin
- Normal Nucleoli
- Mitoses

I used training obtained by Dr. William H. Wolberg, University of Wisconsin Hospitalsⁱ. Using these attributes I constructed a RBF network to perform classification. The two classes were either malignant or benign.

This program is meant to be very versatile and extendable. This same program can be used to perform either classification or regression on a wide array of data sets. If you wish to use a RBF network only changes to the [script.xml](#) file are necessary. If you would like to create your own model, you can easily add a model class to the C# source code. I tested this program on several data sets from the University of California at Irvine Machine Learning Dataset Repositoryⁱⁱ.

Using the Program

I included both the source and compiled forms of this program. Additionally I included the data file for the breast cancer research. This program operates on Excel .XLSX files. To take the program through its paces, use the following steps.

1. Launch the GeneticAlgorithmUtil.exe application.
2. Fill in the location of the script.xml file that you would like to use. If you are running from the folder I sent, it will most likely find the included script.xml file.
3. Click **Load** to load the script.
4. Click **Generate** to generate and score an initial random population.
5. Click **Train** to train the population. The program will show the number of child genomes created and the current best score. Stop training when the score goes below 0.01. This can take a few minutes, depending on your computer's speed. It usually takes me around 70k genomes to get to 0.01.
6. At this point you have a trained population, you can **Save** it if you wish.
7. Now lets evaluate it on data not part of the training set. Click **Evaluate**.
8. A file named Output.xlsx will be generated. You can see the predictions from the program. It should be very accurate.

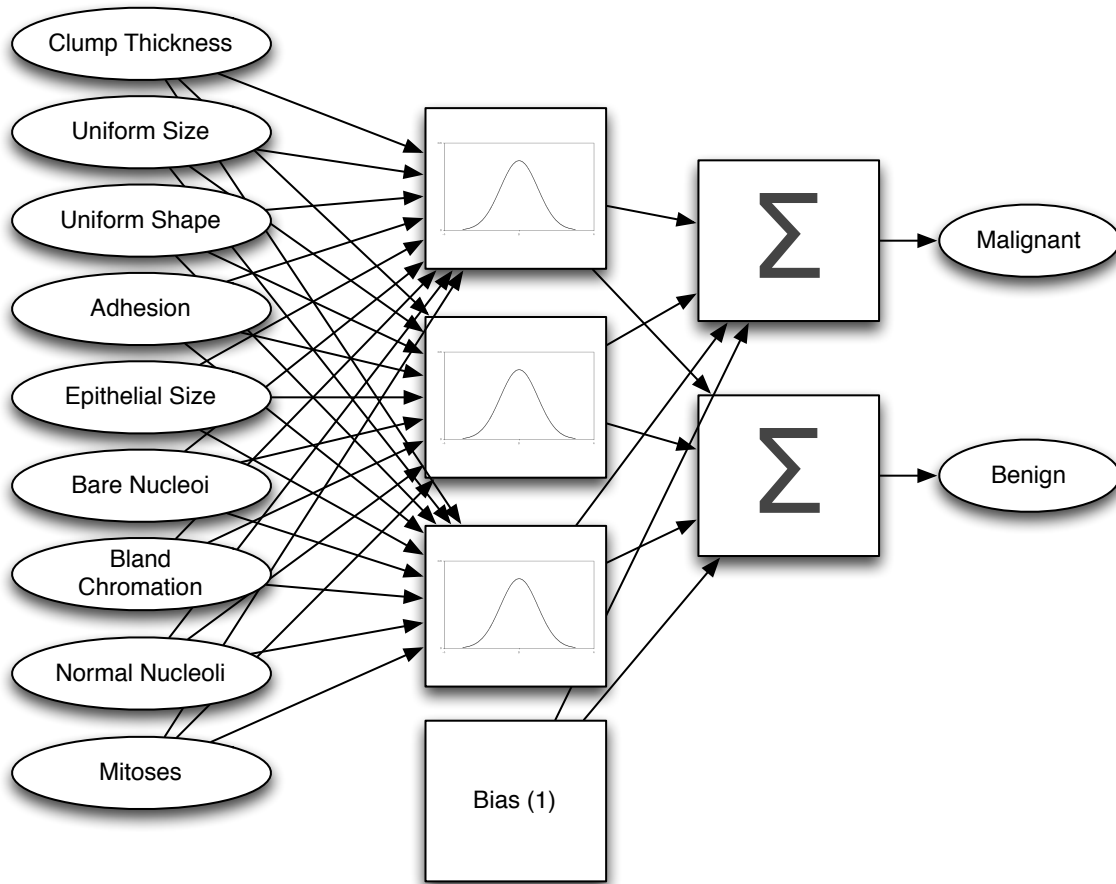
Radial Basis Function Network Description

A Radial Basis Function network is a statistical model that can be used for both classification and regression. It provides a weighted summation of one or more Radial Basis Functions, each of which receives the weighted input attributes used to predict. The following equation summarizes a RBF network.

$$f(X) = \sum_{i=1}^N a_i p(\| b_i X - c_i \|)$$

Where **X** is the input vector of attributes, **c** is the vector center of the RBF, **p** is the chosen RBF (i.e. Gaussian), **a** is the vector coefficient (or weight) for each RBF, and **b** species the vector coefficient to weight the input attributes.

Graphically this can be viewed as follows.



Arrows represent all coefficients from the equation. The arrows between the input attributes and RBFs are represented by **b**. Similarly, the arrows between the RBF's and the summation are represented by **a**. You will also note that there is a bias box. This is a synthetic function that always returns 1. Because the bias function's output is constant, no inputs are required to it. The weights from the bias to the summation function very similarly to the vector intercept in linear regression.

Because there are multiple summations, you can see that this is a classification problem. The highest summation specifies the predicted class. If this were a regression problem, there would be single output. This single output would represent the predicted output for regression.

Genetic Algorithm Description

A Genetic Algorithm typically evolves a population of potential solutions to a problem. Each potential solution is typically called a Genome or Chromosome. Each potential solution is represented as a "DNA Strand". This DNA Strand is an array of numbers. To evolve a RBF I treat each RBF as an array of numbers. These arrays contain the following values:

- Input coefficients
- Output/Summation coefficients

- RBF Width Scalars (same width in all dimensions)
- RBF Center Vectors

RBF networks are typically trained either using Gradient Descent or Matrix Transformations. However, such approaches only adjust the coefficients. This leaves the RBF parameters to manual selection.

Using a Genetic Algorithm allows me to evolve all parameter to the RBF network. The only aspect that I am not evolving is the number of RBF functions to use. This must be define in the script.xml file. This is because most Genetic Algorithms use a fixed DNA size. This is biologically plausible because mating only occurs through genetically similar genomes (i.e. those of the same species).

Genetic algorithms require a score function to determine the superior genomes. The scoring function for this program is very simple. The training data is used to see how accurately a given genome (RBF network) can predict a given malignant or benign status.

This Genetic Algorithm employs many advanced techniques from current research.

- **Fully Multithreaded** – This program makes use of the C# Parallel class to ensure that all available processors and cores are fully utilized. This results in very fast training on modern Multi Core CPU's.
- **Non-Epoch Based** – Unlike many GA's the population is not rebuilt each epoch. Rather parents are chosen from the population and resulting children replace weaker genomes. This is very efficient for multi-threading. It is also biologically plausible. We do not have clear-cut lines of when each generation begins and ends in the real world.
- **Tournament Selection** – When a parent must be chosen, for mutation or crossover, we choose a random member of the population and then try 10 more random members to get a more suited potential parent. When unfit genomes must be removed, this process is run in reverse. This is very efficient for multi-threading, and is also biologically plausible. You don't have to outrun the fastest tiger on earth, just the tigers you randomly encounter on a given day! This also removes the need for a common GA technique, know is elitism.
- **Spectated Crossover** – Crossover is tricky, and can sometimes even under-perform mutation. In the real world successful mating only occurs within a species (actually, that is the definition of species).
- **More than Two Parent Crossover** – Why not have more than two parents? A child can be created from several optimal parents. This is a very interesting technique that I first learned about from a Forecasting & Futurism by Dave Snell.ⁱⁱⁱ

Extending the Program

A Genetic Algorithm can be used to optimize DNA for more than just RBF networks. To facilitate other models simply create a C# class that implements the following methods.

void Init(ConfigScript script, string config);

The Init method simply gives you access to the config.xml script, as well as an optimal configuration string passed to your program. For the RBF network model this is the number of RBF functions.

Genome GenerateRandomGenome(Random rnd, ConfigScript script);

The GenerateRandomGenome method is called to generate a new random genome. This is typically performed as part of the initial population generation.

double[] Compute(double[] input, Genome genome);

The compute method computes the genome's output based on the given input.

double Score(Genome testSubject, double[][] training, double[][] ideal);

The score method determines how fit the genome is. Optional training data can be passed as well.

Genome Mutate(Random rnd, Genome genome);

Mutate the specified genome and return a child. The original genome is not changed.

Genome[] Crossover(Random rnd, Genome[] parents);

Perform a genetic cross over, based on the parents. The children are returned.

ⁱ O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

ⁱⁱ Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.

ⁱⁱⁱ Snell, D: Genetic Algorithms—Useful, Fun and Easy, Page 7, Forecasting & Futurism December 2012.