Darwinian Model Building

Do Kester* and Romke Bontekoe†

*SRON Netherlands Institute for Space Research,
Landleven 12, 9747 AD Groningen.
email do@sron.nl
†Bontekoe Research, Rooseveltstraat 4d, 2321 BM Leiden.
email: romke@bontekoe.nl

Abstract. We present a way to generate heuristic mathematical models based on the Darwinian principles of variation and selection in a pool of individuals over many generations. Each individual has a genotype (the hereditary properties) and a phenotype (the expression of these properties in the environment). Variation is achieved by cross-over and mutation operations on the genotype which consists in the present case of a single chromosome. The genotypes 'live' in the environment of the data. Nested Sampling is used to optimize the free parameters of the models given the data, thus giving rise to the phenotypes. Selection is based on the phenotypes.

The evidences which naturally follow from the Nested Sampling Algorithm are used in a second level of Nested Sampling to find increasingly better models.

The data in this paper originate from the Leiden Cytology and Pathology Laboratory (LCPL), which screens pap smears for cervical cancer. We have data for 1750 women who on average underwent 5 tests each. The data on individual women are treated as a small time series. We will try to estimate the next value of the prime cancer indicator from previous tests of the same woman.

Keywords: Evolutionary Programming, Model Selection, Nested Sampling, Time Series, Cervical Cancer

THE DATA

In the Netherlands every woman between 30 and 60 years old, is invited to participate in a free test to detect cervical cancer, every 5 years. Such a pap smear tests yields 2 values, an O-value and a P-value. The P-value measures the status of the squamous epithelium, and it is an indicator for stages in cervical cancer development. The O-value measures inflammatory events; it consist of one from a set of 9 unrelated conditions (bacterial, viral or fungal infections). Depending on the P-value diagnosis, an additional high risk Human Papilloma Virus (HPV) test is performed. Therefore only a small fraction of the women in LCPLs database of pap smear tests also have HPV tests. The only other piece of information the we have about these women is the age at which the tests were done. A list of the conditions for each of the P- and O-values is given in Table 1. In practice women for whom a P-value of 5 or higher is obtained are sent to a gynaecologist for further treatment.

From the 300000 women in the database we selected those for which at least 2 HPV tests could be found. As a consequence every pap smear test has an associated HPV value: not done, positive or negative. The selection resulted in case histories for about 1750 women that comprise on average 5 tests. Half of the case histories we used to build our models, the other half to test the final result.

TABLE 1. The conditions for O-values and P-values as listed in Roeters et al. [3]. The O-values are unrelated and disjunct. The P-values are increasing in severity. They are translated to the Bethesda system.

O-value	O-condition	P-value	P-condition
1	Koilocytosis	1	Normal
2	Trichomonas	2	ASCUS
3	Dysbacteriosis	3	ASCUS
4	Candida	4	LSIL
5	Gardnerella	5	HSIL
6	Healthy	6	HSIL
7	Actinomyces	7	HSIL
8	Chlamydia*	8	Carcinoma
9	Leucocytosis	9	Carcinoma

* This identification is now rejected. The exact cause is unknown.

The aim of this paper is to investigate whether we can predict a womans next P-value from her previous tests. The case histories are treated as a set of short time series.

MODEL

We want to construct a model which can predict the next P-value given the results of previous tests on the same woman. One model is applied to all case histories. As there is no theory we can follow, we develop heuristic models that we let evolve according to the Darwinian principles of variation and selection. Borrowing terminology from biology, each model has a genotype and a phenotype. Variation is working upon the genotype, while the selection kicks in at the phenotype level.

Genotype

The genotype consists of one chromosome containing one gene. Our model is extremely simple compared to whatever exists in biology. The gene is a linear string of primitives that we call bases, again borrowing from genetic biology.

Bases can be various data values, model parameters, operators, simple functions and memory locations. Each base is assigned an ASCII character. A chromosome consists of a string of characters. The string is interpreted as a program written in Reverse Polish Notation (RPN) also known as postfix notation. In Table 2 we define the set of bases used in this paper. Now the RPN string `aYb+*R` reads as $\sqrt{a\,(Y+b)}$ in normal notation.

In RPN a stack is defined to do the calculations. The 'count' column in Table 2 indicates how the stack changes for every base. Valid genes must have a stack count of 1. Sequences of bases form a 'base term' if the stack count equals 1. So a single parameter is a base term, as is e.g. `aX+`. We limit chromosome strings to 80 characters. With these definitions the accessable model space, $\mathfrak{M}$, contains about 10^{60} valid models.

TABLE 2. The set of bases

base	type	count	description
a b c d e f g h	real	+1	parameters, θ, to be estimated
X	real	+1	age (in decades) at the time of the test
Y	real	+1	time to the next test
Z	real	+1	P-value
B C D E F G H I J	boolean	+1	O-values, only one is true
K	integer	+1	HPV-value
+ - * /	operator	-1	arithmetic operators: add, subtract, multiply, divide
< > $\sim$	operator	-1	relational operators: smaller, larger, equal
& |	operator	-1	boolean operators: and, or
?	operator	-2	if-operator: abc? reads as if(c) then b else a
S L R A	function	0	functions: sign, log, sqrt, arctan
p q m	real	+1	read from memory location
P Q M	null	0	write to memory location

All functions and the division must be redefined such that they do not crash when they are called with arguments outside the valid domain. We choose to return a 0 for all functions with out-of-domain arguments so `Xa*b+R` is in ordinary notation $\sqrt{\max(aX+b,0)}$. Other choices might be equally valid. Since Darwinian programming visits every accessable nook and cranny, all bases must be covered.

Memory locations always come in pairs, one for writing and one for reading. The order of reading and writing determines their use. If the writing is done before the reading in the gene, then the stack value at the time of writing is stored and it is retrieved when the reading operation occurs. It is copying a stack value to a later moment in the processing. If the reading is done before something is written, the value retrieved is what was stored during the previous cycle in the time series. This way information can be passed from one measurement to the next. Of course there is no value to read at the start of every time series. We solve this by treating this first value as an extra free parameter of the model. These parameters, just like the others, apply to all time series.

In the literature on genetic programming [2], tree structures are favored instead of chromosome strings. Of course they map one to one on each other. We regard strings as a simpler representation which have the added bonus that they look more like their biological counterparts. Contrary to common usage in genetic programming where the numeric values of model parameters are evolved along with the model itself, we use parametrized models. The optimal parameters for the models are found in a more conventional way. This way we have a more thorough search through the model space, $\mathfrak{M}$. And it opens the possibility to use Bayesian evidence to discriminate between competing models.

Variation

Darwinian variation is acting on the genotype level. It is a way to generate a new model from one or more existing ones. We define six variational operations in Table 3.

TABLE 3. The list of Variation Operations

	operation	description genes	result
1	cross-over	Glue the head of one gene to the tail of another.	
		`aXb*/Zc*/J/`	`aXb*/Z*Z+YY/pA-P*`
		`Za+p/b*Z+YY/pA-P*`	
2	insert	Splice part of a gene into another.	
		`aaJI|?b+ZS*`	`aaZL/JI|?b+ZS*`
		`abZL/-c*d*`	
3	mutate	Change a base into a similar one.	
		`aZRJY+LLL//b*Z+`	`aZRJY+LLA//b*Z+`
4	add	Insert a base (and operator) into a gene.	
		`aI/`	`aX*I/`
5	delete	Remove a base (and operator) from a gene.	
		`aX*I/`	`aX*`
6	memory	Insert a memory pair.	
		`JaYb*//c*`	`Jap-Yb*//Pc*`

The operations 1 to 5 are common practice in genetic programming [2]. They deal predominantly with local changes. The sixth variational operation facilitates the occurrence of memory pairs, a read and a write, at different locations.

In all operations the stack count is conserved, guaranteeing a valid algorithm.

All newly created chromosomes undergo various cleaning and simplifying operations. This is done to reduce the amount of introns, pieces of code that have no influence on the end result.

Phenotype

The phenotype is how the genotype (or model, M) manifests itself in the environment. In our case the environment is represented by the data. One genotype generally represents a whole family of related models, defined by the actual values for model parameters, θ. We first optimize the model parameters and determine the evidence given the data. The model, $M(\hat{\theta})$, which carries the highest evidence is selected as the phenotype for a given genotype.

For parameters θ, data D and a model M we write Bayes theorem as

$$\begin{aligned} \Pr(\theta D|M) &= \Pr(\theta|M)\times\Pr(D|\theta M) &= \Pr(D|M)\times\Pr(\theta|DM) \\ \text{joint} &= \text{prior}\times\text{likelihood} &= \text{evidence}\times\text{posterior} \end{aligned} \tag{1}$$

In our case we know next to nothing about the models or its parameters. But as all our data have values between 0 and 10, we take the priors for the parameters uniformly within [-100,100].

The algorithm we use is Nested Sampling [5, 4] with automatic noise scaling. It starts with an ensemble of N members with values for θ, randomly chosen over the prior domain of the parameters; we normally take N = 100. For each of the ensemble members

the likelihood is calculated. An iterative cycle is started where the member with lowest likelihood is replaced by a copy of one of the others. The parameters of the copy are randomly moved around over the prior, provided that the new likelihood remains larger than the original one. This way we slowly climb the likelihood mountain. The evidence follows as a integral of the likelihood, while the optimal parameters are weighted sums over all the points visited. This is standard Nested Sampling practice.

The random movements and the continuous removal of worst points, makes Nested Sampling a genetic (or Darwinian) algorithm in it own right. We call this stage level 1 of Nested Sampling.

Compilation

The many thousands of iterations needed in one Nested Sampling calculation call for an efficient way to calculate the model function. The simplest method is an interpreter which reads the chromosome string, character by character and acts accordingly. It is easy and attractive but much too slow.

Nordin and Banzhaf [1] and others [2] report about directly evolving machine code, giving speedups of a factor 100. The difficulty is that it is hard to interpret what the actual result of the evolution is. We would rather stick to a more conventional record of what the model is doing.

We found a different option to obtain optimized computational speeds using shared (or dynamic) libraries. Shared libraries can be linked and unlinked in a running program. The chromosome strings are translated into a C funcion. The function is compiled and linked into a shared library, which is linked and unlinked runtime when it is needed. An example of a (annotated) translation is given in Figure 1.

Selection

Model selection uses a second level of Nested Sampling. On level 1 we calculate the evidence, a measure how well the data fit the model. This evidence we can use in a second level of Nested Sampling, now to find better models. Again we write Bayes theorem, this time on the level of the models, M

$$\Pr(MD|\mathfrak{M}) = \Pr(M|\mathfrak{M}) \times \Pr(D|M\mathfrak{M}) = \Pr(D|\mathfrak{M}) \times \Pr(M|D\mathfrak{M}) \tag{2}$$

where $\mathfrak{M}$ is the set of all accessable and valid models. On this second level the likelihood is the same entity as the evidence on level 1, viz. $\Pr(D|M\mathfrak{M}) = \Pr(D|M)$ which is logically correct because M is a member of the set $\mathfrak{M}$.

A non-uniform prior on the models, $\mathfrak{M}$, is required. When two models perform exactly the same, the shorter one should be preferred. However real innovation should not be suppressed, just because its genotype is longer. After some experimenting we settled on $\Pr(M|\mathfrak{M}) = \exp(-0.1L)$, where L is the length of the chromosome.

```c
/*
 * Automatically generated function by ga_make_function
 *   Copyright Do Kester, Zuidhorn  2010
 *
 * Original gene (genotype):
 *   pqa*L+K-RZ-ALKDIb*/-c*d+*AeX/I+*LLQfX/QZ*PR-S
 */

#include <math.h>
#include "user.h"

#define sqrt( x )       ( ( (x) >= 0 ) ? sqrt( x ) : 0 )
#define log( x )        ( ( (x) > 0 ) ? log( x ) : 0 )

double ga_function052(
    UserStr     *user )         // user struct
{
    int         i, k, K, D, I, *CVK;
    float       X, Z;
    double      a, b, c, d, e, f, p, q;
    double      aa, bb, cc, dd, zz;
    double      chisq = 0;

    a = user->param[0];         // parameters
    b = user->param[1];
    c = user->param[2];
    d = user->param[3];
    e = user->param[4];
    f = user->param[5];
    CVK = user->categ[0];
//  loop over all women
    for ( k = 0; k < user->data->length; k++ ){
        p = user->param[6];     // reset memory
        q = user->param[7];     // reset memory

//      loop over times series per woman
        for ( i = 0; i < user->data->nr[k]; i++ ) {
            X = user->data->f1[k][i];       // age
            Z = user->data->f3[k][i];       // P-value
            D = user->data->b3[k][i];       // O-3
            I = user->data->b8[k][i];       // O-8
            K = CVK[user->data->c1[k][i]];  // HPV

//          the algorithm starts here
            aa = q  * a  ;
            aa = log( aa );
            aa = p  + aa ;
            bb = aa - K  ;
            aa = sqrt( bb );
            bb = aa - Z  ;
            aa = atan( bb );
            aa = log( aa );
            bb = I  * b  ;
            bb = ( bb ) ? D  / bb : 0;
            bb = K  - bb ;
            cc = bb * c  ;
            dd = cc + d  ;
            bb = aa * dd ;
            aa = atan( bb );
            bb = ( X  ) ? e  / X  : 0;
            cc = bb + I  ;
            bb = aa * cc ;
            aa = log( bb );
            aa = log( aa );
            aa = q + aa ;
            bb = ( X  ) ? f  / X  : 0;
            bb = q - bb ;
            cc = bb * Z  ;
            bb = p - cc ;
            bb = sqrt( bb );
            bb = aa - bb ;
            aa = -( bb );

            user->data->mock[k][i] = zz = aa;  // result
            zz = zz - user->data->out[k][i];   // residual
            chisq += zz * zz;
        }
    }
    return chisq;  // sum of squared residuals
}
```

FIGURE 1. Annotated C-translation of a chromosome string

Evolution

With the prior and the evidence we can play the Nested Sampling game at a second level to find better models from the $\mathfrak{M}$-space. Its implementation is however not as simple as on the first level.

When in level 1 the member with the lowest likelihood is replaced by one of the others and the parameters have to be moved around to find a new point, we can assume continuity on the part of the parameters, θ. Somewhere in the neighbourhood, far or near, that new point always can be found.

On level 2, applying a variation operation from Table 3, yields a completely different model with a different evidence. When this evidence is lower than before, the variation is rejected and a new variation must be tried. The concept of continuity in models from $\mathfrak{M}$ does not exist. Consequently increasingly more trials are needed before we find a model that remains within the likelihood constraints of the Nested Sampling algorithm.

We also find that not all variation operations are equally efficient, and their efficiency changes over time. In level 2 we reassess every 100 cycles the efficiency and adapt the relative frequencies with which the operations are chosen. In Figure 2 we display the efficiency of the various variation operations. In general the efficiency goes down.

Contrary to received wisdom the cross-over operation is not the most efficient one. Especially later in the process its efficiency drops below that of the simpler add, delete and mutate operations. Presumably the simpler operations have more chance of yielding a close-by model.

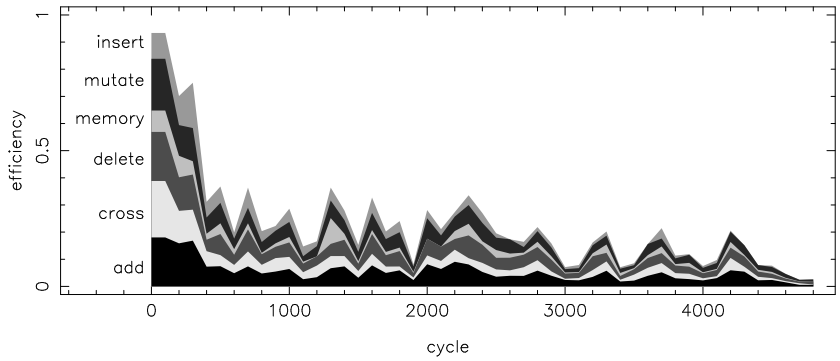

FIGURE 2. Efficiency of the variation operations of Table 3 over time

RESULTS AND CONCLUSIONS

After 100 days of computing and fitting parameters to more than 40000 models, the results for the best model is shown in Figure 3. We display the data in the control set; the test set itself yields virtually the same figure.

In Figure 3 the model is presented by re-ordering the data. The order is determined by the predicted P-values which is the line in the lower panel. The dots represent the actual P-values. Note that the data are *not* in time order in this figure, not even for individual women. The middle panel displays the corresponding O-values. Only values which are actually present in the chromosome string are shown. The top panel displays the HPV-coding, where the value of 0 is no measurement, 1 is a negative HPV test and 2 is a positive test.

The string at the top is the chromosome string which represents the model.

The detailed agreement between the predicted and actual P-values is not very great. However overall, there are some interesting points in this figure. In the HPV panel we see that the point with HPV = 1 (negative test) are concentrated to the left while the points with HPV = 2 (positive) are mostly to the right. This is in agreement with the medical knowledge that the presence of HPV is associated with the occurrence of cervical cancer.

In the O-value panel we see that the points with an O = 1 (Koilocytosis) are concentrated to the right. It is known that Koilocytosis is associated with the occurrence of HPV. More carefull inspection of again the O-value panel shows that the O = 8 shows also a slight concentration to the right. The exact cause of O = 8 is still unknown. It has been suggested that it might be a precursor or a variant of O = 1, which would agree with what we find.

The results are maybe a bit disappointing in the sense that little new is creeping out of the woodworks. On the other hand all known associations with high P-values are found. In that aspect the Darwinian evolution did work as expected. It could very well be that the information we are seeking is just not present in the dataset we have.

At the convergence of level 2 Nested Sampling has reached a compression factor, $H = 23.9$. On level 1 the compression factor was 20.9. This can be interpreted that the best model represents 1 out of $\exp(44.8)$. As we visited 40000 models to find this model, we can conclude that Darwinian exploration is 7×10^{14} times more efficient than random trials.

DpKZa*++blJ|?D+K?PZ*LE+LpB+*LpY-KZRG-a**Y-+I+K?PZq+Za*+aQ+K//bQ+Z*J-K+QX/R

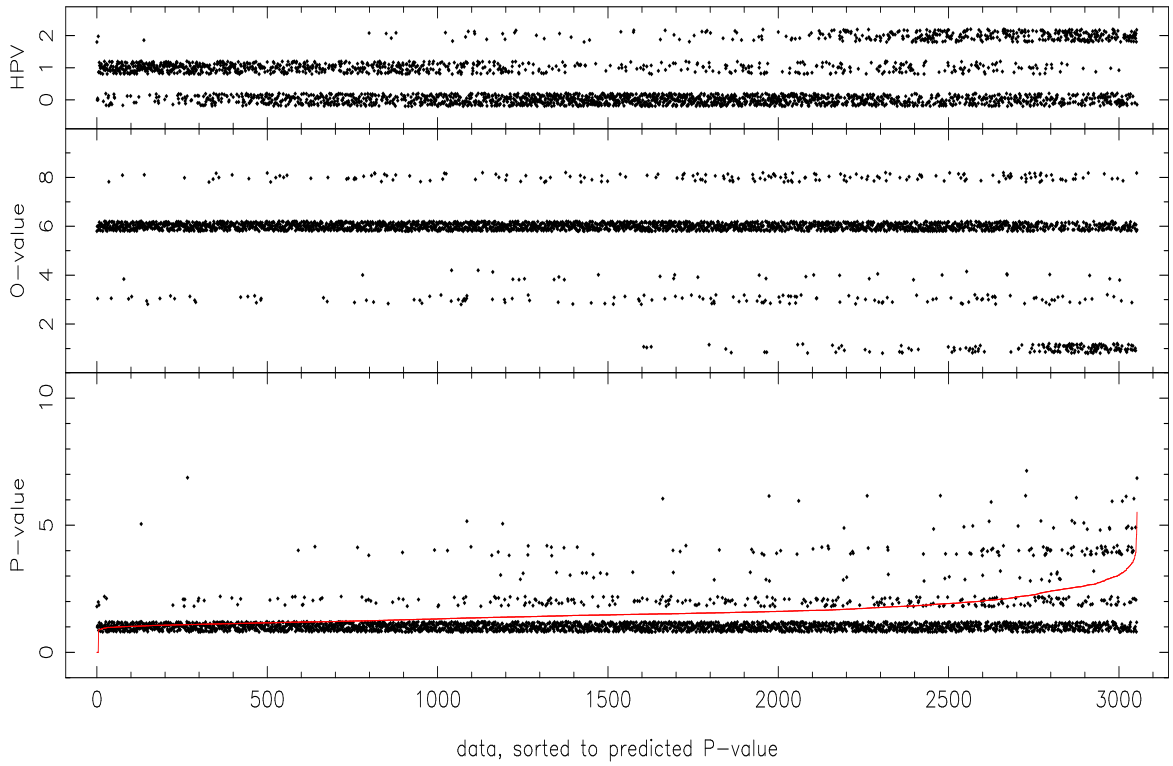

FIGURE 3. The best model found. All integer values are randomized a little to better show the density.

ACKNOWLEDGMENTS

We would like to thank Dr. Mathilde E. Boon, director of the LCPL, for permission to use their (anonymized) data and for enlightening discussions about the medical aspects of the data.

REFERENCES

1. P. Nordin, and W. Banzhaf, "Programmatic Compression of Images and Sound," in *Genetic Programming* edited by R. Koza et al., MIT Press, Cambridge, 1998, pp 345-350.
2. R. Poli, W.B. Langdon, and N.F. McPhee, *A Field Guide to Genetic Programming* Published via http://lulu.com and http://www.gp-field-guide.org.uk, 2008.
   This is a recent overview of the field with lots of references inside, publicly available under the Creative Commons License.
3. A.M.E. Roeters, M.E. Boon, M. van Haaften, F. Vernooy, Tj.R. Bontekoe, and A.P.M. Heintz, "Inflammatory Events as Detected in Cervical Smears and Squamous Intraepithelial Lesions," in *Diagnostic Cytopathology*, Vol 38, No 2, pp 85-93, 2009.
4. D.S. Sivia, and J. Skilling, "Data Analysis, A Bayesian Tutorial, 2nd Edition," Oxford University Press, 2006.
5. J. Skilling, "Nested Sampling for General Bayesian Computation," in *Bayesian Analysis 4*, 2006, pp 833-860.