University of Essex

Evolving Regular Expressions for GeneChip Probe Performance Prediction

Presented at PPSN 2008

W. B. Langdon[1,2] and Andrew Harrison[1,2]

[1]Department of Biological Sciences and [2]Department of Mathematical Sciences, University of Essex (Email: wlangdon@essex.ac.uk)

A strongly typed grammar based Genetic Programming implementation written in gawk and using linux egrep

1 Introduction

Affymetrix HG-U133A GeneChips (Fig. 2) typically provide 11 redundant measurements. These should be correlated (Fig. 1). Sometimes they are not. Sometimes this may be due to the DNA sequence. STGP is used to find regular expressions to classify poor DNA sequences.

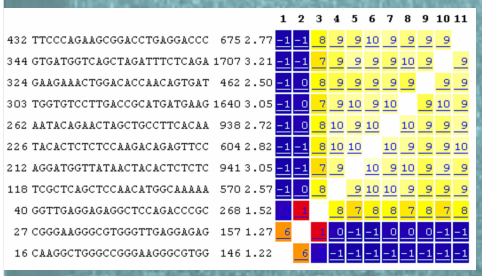

Figure 1: 200660_at.pm, S100 calcium binding protein A11. Correlation between 2757 HG_U133_Plus_2 Human tissue samples (RNAnet bioinformatics.essex.ac.uk).

2 Methods

How are poor probes identified?

Thousands of human GeneChip CEL files taken from GEO, quantile normalised and spatial flaws removed. Correlations for all probes in all HG-U133A probesets calculated (e.g. Fig. 1). Use only 4118 probesets with strong signal (i.e. ≥10 non-overlaping probe pairs ≥0.8). 583 probes not correlated with rest of probeset. 7832 strongly correlated with other probes.

Grammar based GP gives egrep pattern

4118 probesets randomly split in 3 (train test, verification). BNF grammar (Fig. 3) gives legal, regular expression, which can be executed by unix egrep to give number of DNA probe sequence which it matches. GP fitness (8.5mS) given by number positive examples matched versus number negative examples matched.

Next generation by sort and gawk

1000 fitness values **sort**ed and top 200 give breeding pool for next generation (Fig. 4). First crossover point chosen from decision rules (in blue). Strong typing limits **gawk** to chose same rule as second crossover point.

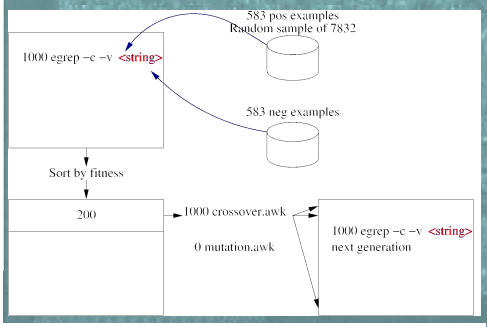

Figure 4: Population is unix command script. Fitness of each regular expression given by egrep -c

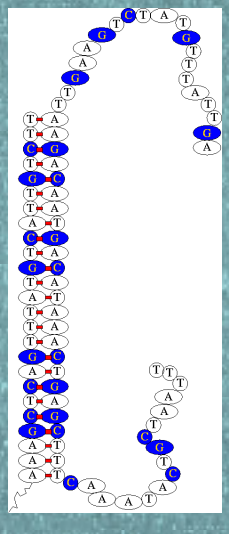

Figure 2: Example of Affymetrix probe (left) binding to fluorescent mRNA gene fragment (right) 209649_at.pm5 binds to signal transducing adaptor molecule (SH3 domain and ITAM motif) 2.

3 Results

Figure 5: Genotype of best of generation 50: GC{3}|G{4}|C{4}|CG{1}C{2}|GG{4}C+|G(G|C){4}|G(G|C){4}|C{3}.
It is equivalent to GGGG|CGCC|G(G|C){4}|CCC}.

GGGG		Evolved	
≤0.3	>0.3	≤0.3	>0.3
209	434	**448**	4436
413	14047	174	10045

Table 1: confusion matrix on verification data. Evolved regular expression finds more than twice as many poor probes than human GGGG rule.

4 Discussion

Binding of RNA to cDNA in solution relatively well understood but the binding of mRNA fragments to cDNA probes on arrays surface poorly understood. Datamining offers an additional entry point.

Is poor correlation due to non-specific binding?
Do both GGGG and CCC (Fig.5 and Tab. 1) provide nucleation sites for any mRNA to bind, and this non-gene specific mRNA is not fully washed away?

Acknowledgements
I would like to thank Jo Rowsell.

```
<start> ::= <RE>
<RE> ::= <union> | <simple-RE>
<union> ::= <RE> "|" <simple-RE>
<simple-RE> ::= <concatenation> | <basic-RE>
<concatenation> ::= <simple-RE> <basic-RE>
<basic-RE> ::= <RE-kleen> | <elementary-RE>
<RE-kleen> ::= <minmaxquantifier> | <kleen>
<kleen> ::= <star> | <plus>
<star> ::= <elementary-RE2> "*"
<plus> ::= <elementary-RE2> "+"
<minmaxquantifier> ::= <elementary-RE4> "{" <int> <optREint> "}"
<elementary-RE> ::= <group> | <elementary-RE1>
<elementary-RE1> ::= <xos> | <elementary-RE2>
<elementary-RE2> ::= <any> | <elementary-RE3>
<elementary-RE3> ::= <set> | <char>
<elementary-RE4> ::= <group> | <elementary-RE2>
<group> ::= "(" <RE> ")"
<xos> ::= <sos> | "$"
<sos> ::= "^" <elementary-RE4>
<set> ::= <positive-set> | <negative-set>
<positive-set> ::= "[" <set-items> "]"
<negative-set> ::= "[^" <set-items> "]"
<set-items> ::= <set-item> | <set-items2>
<set-items2> ::= <set-item> <set-items>
<set-item> ::= <char>
<char> ::= <c00> | <c01>
<any> ::= "."
<c00> ::= T | C
<c01> ::= A | G

<optREint> ::= <2ndint> | $
<2ndint> ::= "," <int>
<int> ::= <d0>
#4 Bit Gray Code Encoder
<REdigit> ::= <d111> | <d0>
<d0> ::= <d00> | <d01>
<d00> ::= <d000> | <d001>
<d01> ::= <d010> | <d011>
<d000> ::= 1
<d001> ::= 3 | 2
<d010> ::= 7 | 6
<d011> ::= 4 | 5
<d111> ::= 8 | 9
```

Figure 3: BNF syntax for egrep DNA sequences. Blue decision rules. Red phenotype.

5 Future Work

The use of small "motifs" is wide spread in microbiology. Bioinformatics is over running with data, which might serve for training GP. Given a suitable grammar, even a crude GP can, in six minutes, find regular expressions of the sort that biologist are familiar with and frequently use.

Grammar based genetic programming has automatically generated regular expressions which differentiate RNA produced by non-coding genes from protein coding mRNA.

Same gawk system generates code for climbin robot

Many other examples should be possible.

Linux code ftp://cs.ucl.ac.uk/genetic/gp-code/RE_gp.tar

6 Summary

- Public archives of thousands of arrays are now available for many types of GeneChip and several organisms (human, fruit fly, Arabidopsis). Exon, ChIPChip and similar arrays coming onstream.
- Correlation matrices show many probes give unusual values. Some of this may be due to DNA sequence. Some due to location on chip. Some for as yet unknown reasons.
- Using BNF grammar makes it easy for GP to automatically generate in a few minutes patterns which distinguish poor probes and may help to explain chip performance. Leading to better analysis and perhaps better designs.