

Metrics for Measuring GP Performance

- General scenario:
 - For a given selection of learning parameters multiple runs are necessary;
 - Population seeds → establish the contribution of the initial population to convergence
- Case of “toy problems” with a suitably significant likelihood of providing an optimal solution:
 - Computational effort
- Otherwise
 - Rank based and non-parametric statistics (see [1])

Computational Effort [2, 3]

- Objective
 - Let our metric for “Computational Effort” express the *number of fitness evaluations* necessary to yield a solution *satisfying the goal criterion* with a suitably high probability.
- Case #1 – each GP run is successful
 - When each run (unique initialization of the population) results in a solution, it is relatively easy to define the “Computational Effort” following application of an appropriate statistic.
 - 1st, 2nd (median), 3rd Quartile
 - Robust statistic with minimum assumptions regarding the data distribution.
 - Mean and variance
 - Fine as long as the data conforms to a Normal distribution (i.e. you have sufficient points to verify this), which it rarely does!

- Case #2 – some fraction of each GP run is successful
 - If successful individuals are not encountered on run ‘ r ’ by evaluation ‘ i ’ do you continue the run or terminate?¹
 - Consider the following Probabilistic Framework [2].
 - Empirically observe the ‘probability’ that run r with evaluation size S yields a solution by evaluation i .
 - $P(S, i)$
 - For each run, compute the Cumulative Probability of recording a success over a set of runs, $r \in \{1, \dots, R\}$, where each run is a unique initialization of the population, size M .
 - $C(S, i)$
 - I.e. if every initialization results in a solution then $C(S, i) = 1.0$
 - The probability of at least one successful run by evaluation i ,
 - $1 - [(\text{trial 1 not successful}) \text{ AND } \dots \text{ AND } (\text{trial R not successful})]$
 - or, $1 - [1 - C(S, i)]^R$
 - Actually interested in asking the question what is the number of runs necessary to reach a prespecified probability, z , of solving the problem.
 - Let $z = 0.99 = 1 - [1 - C(S, i)]^R$
 - Rearranging in terms of run ‘ R ’ provides the expression,
 - $R \equiv R(S, i, z) = \left\lceil \frac{\log(1 - z)}{\log(1 - C(S, i))} \right\rceil$
 - where $\lceil \cdot \rceil$ is the integer ceiling function (round up).
 - Number of fitness evaluations merely introduces,
 - the number of evaluations over which cumulative probabilities are estimated, i , and;
 - the number of individuals taking place in each evaluation, S .
 - $I(S, i, z) = R(S, i, z) \times S \times i$

¹ The more general term ‘evaluation’ is used to indicate the number of individuals taking part in a fitness evaluation at each epoch. Thus $S = M$ in the case of generational selection and $S = N$ ($N \ll M$) in the case of steady-state tournament selection.

- Computational Effort, E , is *traditionally* defined as the minimal instance, or
 - $E = I(S, i_{opt}, z)$
 - Where i_{opt} is the evaluation instance minimizing $I(S, i, z)$ over any initialization.
- Limitations of the Computational Effort Metric
 - Based on information measured from the experiment,
 - Low number of samples (converging runs) results in several sensitivity problems.
 - I.e. $C(S, i)$ is actually $\frac{\hat{C}(S, i)}{R}$
 - 1) Ceiling Parameter – rounding up actually masks significant variation when the fraction of solutions is ‘low’ [3].
 - 2) Minimizing instance – sensitive to low counts of estimated cumulative probability in which results are clustered around similar evaluation counts [3]. Moreover, minima are naturally biased by outliers and consequently under estimating computational effort.
 - Recommendation
 - Avoid the ceiling operator
 - Summarize the number of solutions, $I(S, i, z)$, in terms of 1st, 2nd (median), 3rd Quartile.

Other Metrics based on solutions found

- Number of terms (functional and terminals) within a solution;
 - Before and after intron removal.
- Percentage of runs resulting in a solution;
 - Provides qualifier to “Computational Effort.”
 - Computational Effort as discussed above can be biased towards a low number of solutions at early evaluation counts, whereas an alternative algorithm might produce a comparatively higher number of solutions at a late evaluation count.

Metrics based on solutions not found

- A large number of problems might never satisfy the fitness criteria for any run.
 - E.g. Classification problems based on real world text corpus;
 - Discovery of rules for data mining;
 - Identification of detection rules for Intrusion detection problems.
- Apply Statistical Metrics (quartiles) to domain specific error metrics of the best case individual from the set of R runs.
 - Example,
 - Intrusion Detection – False Positive and Detection Rates;
 - Classification – Precision and Recall;
 - Sum Square Error – function approximation.
- How might you compare a deterministic machine learning algorithm (SVM, C4.5, Naïve Bayes etc) against an evolutionary method?

References

1. Wineberg M. and Christensen S. (2003) Using Appropriate Statistics. Genetic and Evolutionary Computation Conference. Tutorial Program.
2. Koza J.R. (1992) Genetic Programming – On the Programming of Computers by the Means of Natural Selection. MIT Press.
3. Christensen S., Oppacher F. (2002) “An Analysis of Koza’s Computational Effort Statistic for Genetic Programming,” EuroGP, Lecture Notes in Computer Science, Pp 182-191.

