Real-Valued Schemata Search Using Statistical Confidence

D. Randall Wilson
fonix Corporation
180 W. Election Road
Draper, UT 84020, USA
E-mail: *WilsonR@fonix.com*
WWW: *http://axon.cs.byu.edu/~randy*

Tony R. Martinez
Neural Network & Machine Learning Laboratory
Computer Science Department
Brigham Young University
Provo, UT 84602, USA
E-mail: *martinez@cs.byu.edu*
WWW: *http://axon.cs.byu.edu*

Introduction

Need for Optimization of Real-Valued Vectors

- Weights for radial basis function networks
- Attribute weights for instance-based learning
- Weights for multilayer perceptrons
- Many other problems also require a set of real-valued quantities that yield high accuracy or result in good scores according to some other real-valued evaluation measure.

Genetic Algorithms

- Used to find sets of values without necessarily knowing what the gradient is at each sampled point in the weight space.
- Unfortunately have many sensitive parameters
- Having many homogeneous individuals can waste computation.

“Blocking Race” (BRACE) for Attribute Selection

Moore & Lee [1993]

BRACE Statistical Technique

- Used to decide when sufficient statistical evidence has been gathered to determine whether one “model” is worse than (or at least no better than) another.
- Used to choose between different:
  - learning models
  - parameter settings
  - hypotheses
  - etc.

Attribute Selection—*Finding Binary Attribute Weights*

- *Forward selection* starts with no attributes and tries adding them in a greedy manner.
- *Backward elimination* starts with all attributes and tries removing them.
- Moore & Lee used a *schemata search* to avoid problems with forward selection and backward elimination.
- The BRACE algorithm was used to decide when an attribute could be included or excluded.
- Their algorithm had success in finding such binary-valued weights, but does not support finding real-valued weights.

Real-Valued Schemata Search (RVSS)

• Uses BRACE statistical technique to decide when to narrow search.

• Performs a schemata search on real-valued quantities (called weights, though the technique can be used to search for good settings for almost any real-valued quantities for which there is a realiable evaluation function).

Overview of RVSS Algorithm

• Requires:

  • Training set *T* of *n* training instances.

  • Evaluation function $f(\boldsymbol{w})$ to evaluate a setting of weights.

• Finds: a vector $\boldsymbol{w}$ of *m* weights $w_1 ... w_m$ such that $f(\boldsymbol{w})$ yields as high a value as possible.

• Begin with an initial range $min_i ... max_i$ for each weight (e.g., 0...1)

• Divide each range into *d* subranges (e.g., *d*=2)

• For each iteration

  • Pick a random value for all of the weights within range $min_i ... max_i$

  • Optimize each weight individually:

    • Each subrange of the weight competes in a greedy search, i.e., several values of one weight are tried while holding others constant.

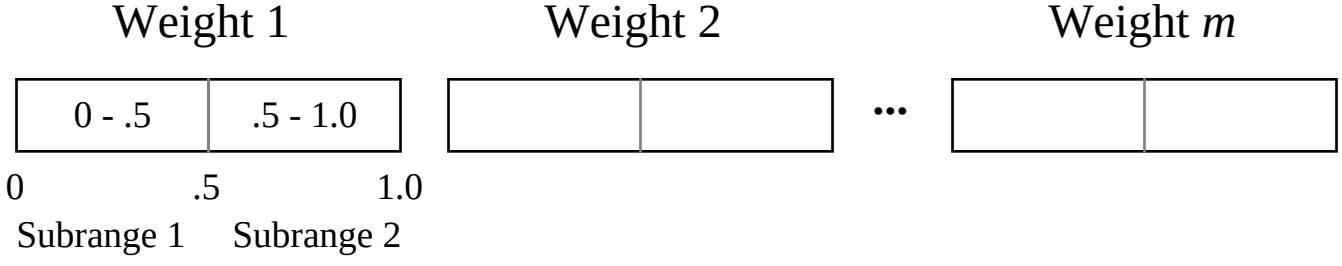
    • If values in a subrange yield lower values for $f(\boldsymbol{w})$ than another subrange, the lower subrange is dropped from the race.

    • If two subranges seem nearly identical, the lower one is dropped.

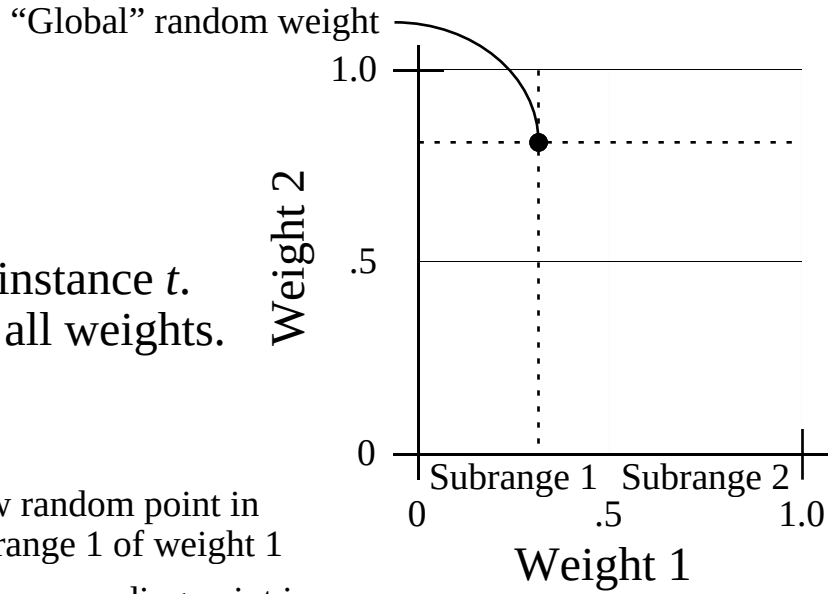
    • If only one subrange remains then the *max* and *min* for the weight are narrowed to cover only the winning subrange.

    • This new smaller range is subdivided again unless the winning subrange is smaller than some threshold *minWidth* (e.g., .0001).

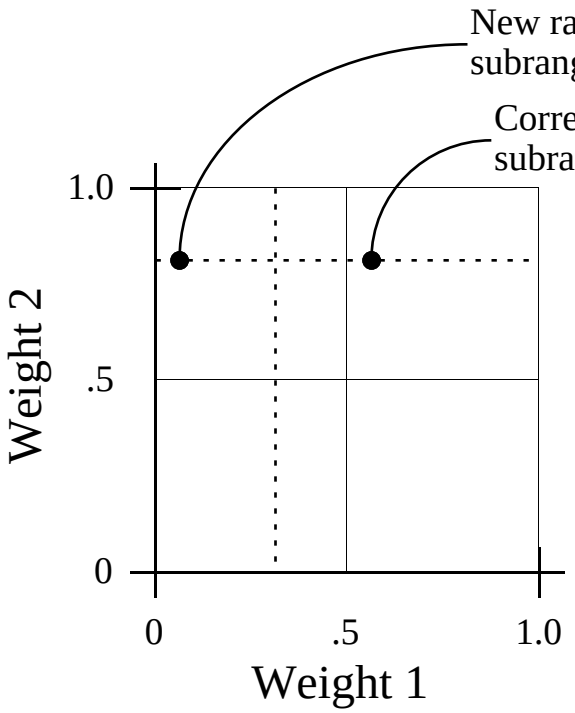
• Once the ranges for all the weights are smaller than *minWidth*, the algorithm is finished.



RVSS

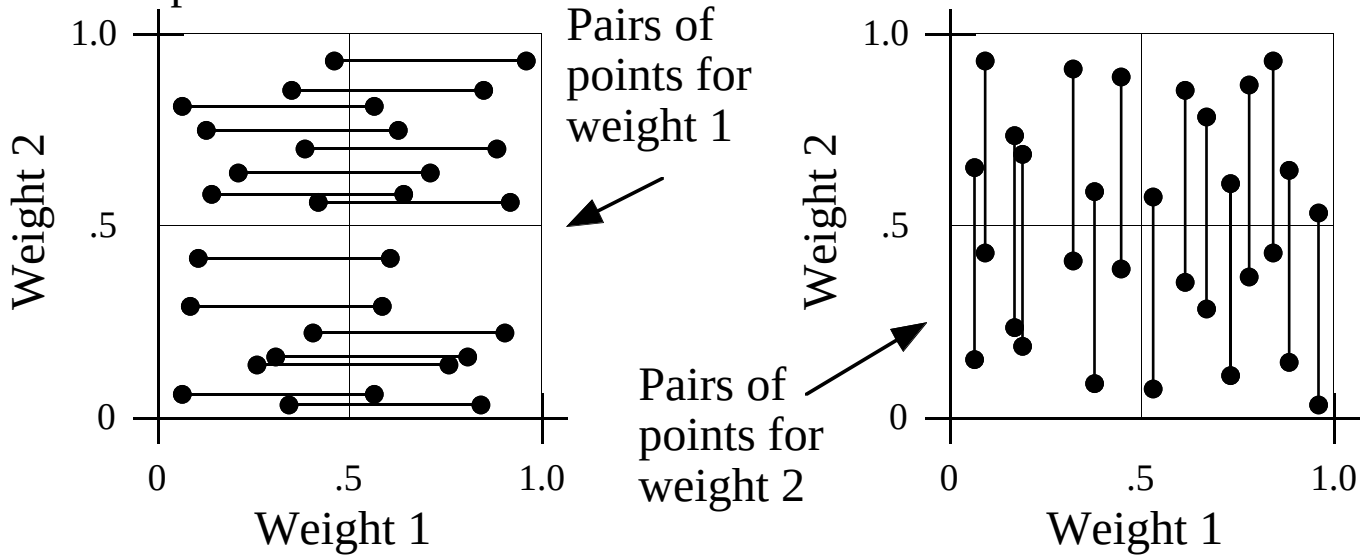


To begin each iteration,
- Choose a random training instance *t*.
- Choose random values for all weights.



Optimize each weight, one at a time.
- Choose a new random point in the first subrange, and evaluate the modified weight vector on *t*.
- Use the corresponding point in all other subranges and evaluate.
- Maintain statistics on the difference between the scores in each subrange.

As time goes on, many corresponding pairs of weights are tested, and statistics are accumulated on the differences between *f*() at the two points.



Statistical Tests

• If one subrange yields significantly lower scores than another, then it can be dropped from the race.

• If two subranges are significantly similar, then the lower one can be dropped from the race.

• Both of these tests are made at once by seeing if:

$$\mathbf{P(}\ \mu < -\gamma\mathbf{)} < \alpha$$

where

• μ is the true mean of the difference between scores yielded by weights in the two subranges

• γ is a small error term

• α is the probability of getting the observed mean by accident.

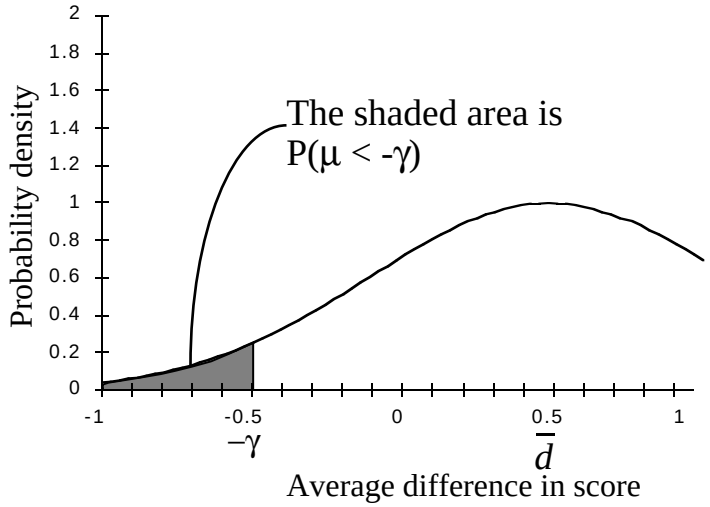
Given two distributions x_i and y_i, $1 \le i \le n$, of two subranges $\boldsymbol{x}$ and $\boldsymbol{y}$, let $d_i = x_i - y_i$ be the difference between observed scores for x and y, and let $\overline{d}$ be the average difference over the n pairs of scores.

If $d > 0$, then $x > y$ on average over the scores observed so far. However, *there is still a chance that y is really greater than x*, i.e., there is still some probability P that the true mean μ of the difference d is negative, (in fact, that it is less than $-\gamma$), even though the average d of the values so far is positive.

By throwing out the lower of the two ranges, there is a therefore a probability P that the truly *higher* range will accidentally be thrown out. Thus, the lower range is not eliminated unless $P < \alpha$, that is, if $P(\ \mu < -\gamma) < \alpha$..

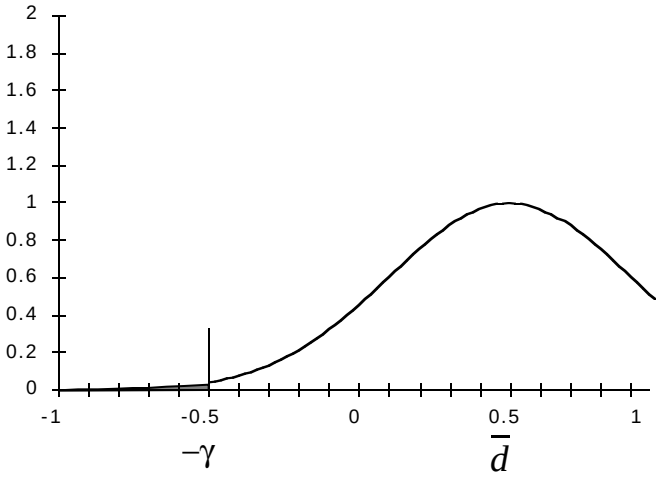
We never want to throw out the higher range, so when $\overline{d}<0$, x and y are switched do make $\overline{d} \ge 0$.

Example 1:



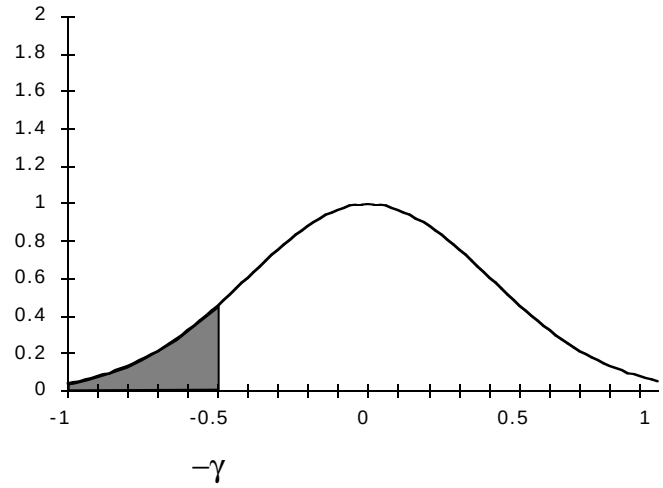
- $-\gamma = -.5$
- $\alpha = .01$
- The average difference $\overline{d} = 0.5$, so $x > y$ on average.
- The variance is large, however, so there is still a reasonable probability (perhaps 10% chance) that y is really greater than x.
- Thus the lower range can not be eliminated without further statistical support.

Example 2:



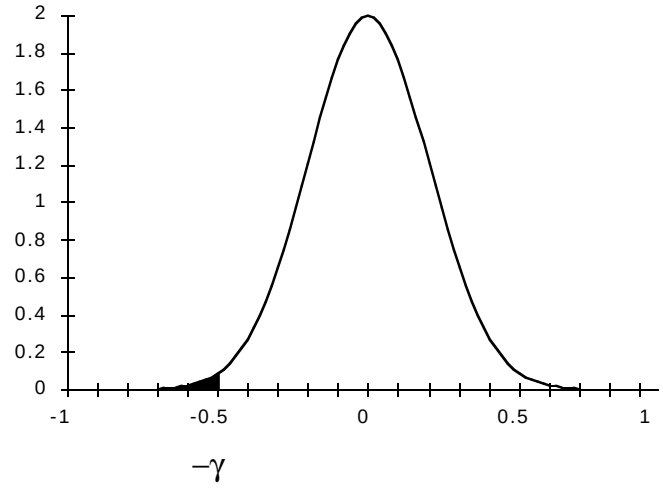
With more data, the variance drops, and the area below $-\gamma$ becomes smaller than α. Since it is extremely unlikely that $x < y$, the lower range y can be dropped.

Example 3:



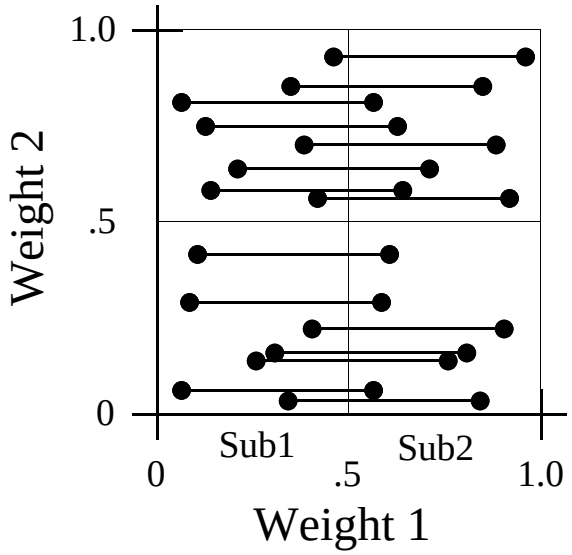
The average difference is 0, and the variance is large, so one subrange may still be better than the other, and we do not know which would be larger, even if they are different.

Example 4:



The average difference is 0, but the variance is small, so there is an extremely small chance that one range is better than the other by more than γ. Thus the smaller range (or either one, if they $\bar{d}=0$ exactly) can be eliminated.

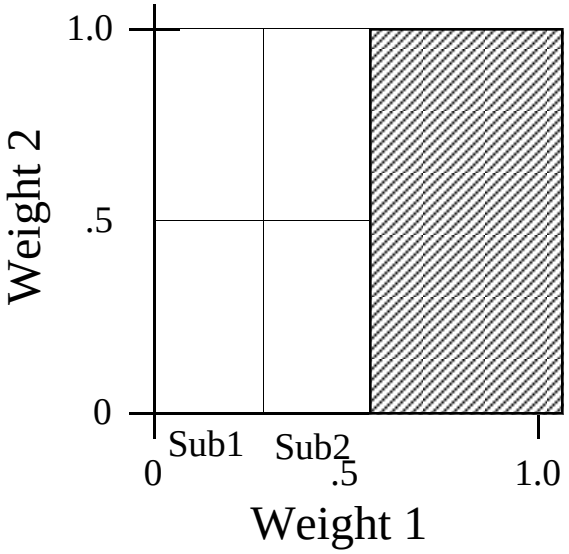
Note that in examples 3 and 4, $P(\mu<0)=.5$ no matter how small the variance gets, so testing for $P(\mu<-\gamma)$ is necessary to eliminate identical subranges.



After several iterations on different training instances, statistical evidence may be sufficient to decide that one subrange can be eliminated from consideration.

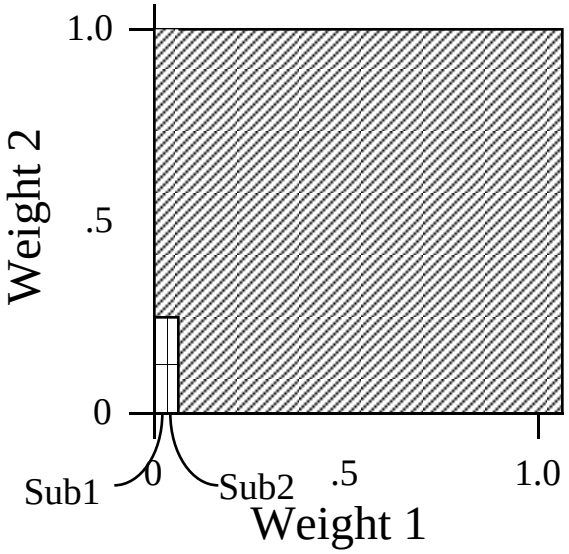
For example, if Sub1>Sub2 (i.e., $d > 0$) and the variance is small, Sub2 can be eliminated.

When only 1 subrange remains, the search space is narrowed to the winning range, which is subdivided into new subranges.



When subranges become small enough for a particular weight, that weight becomes fixed.

When all weights become fixed, the search is complete.



Pseudo-code

```
LearnRVSS(T, alpha, gamma, minWidth):w[1..m]
In:  training set T
     alpha, the probability of an erroneous decision
     gamma, a small allowable error
     minWidth, the smallest width of weight range that is subdivided.
Out: w[1..m]

For i=1..m, r=1..d, and s=r+1..d
     set num[i][r][s], sum[i][r][s] and sumSquared[i][r][s] to 0

While there are still weights left in the race
     Let inst = a random instance from the training set T.
     # Pick random weights:
     For each weight i that is still in the race
          let w[i]=random value in any one of its divisions that is still racing.
     # Test individual weight settings
     For each weight i that is still not permanently fixed
          Pick a random value v in the range 0...1
          For each range r of weight i that is still racing
               Let w[i]=min[i][r]+width[i][r]*v
               Let score[r]=f(w[1..m], inst), i.e., evaluate w[1..m] on training instance 'inst'
          For each range r=1..d of weight i that is still racing
               For each other range s=(r+1)..d of weight i that is still racing
                    Add 1 to num[i][r][s]
                    Add (score[r]-score[s]) to sum[i][r][s]
                    Add (score[r]-score[s])^2 to sumSquared[i][r][s]
                    Let avg = sum[i][r][s] / num[i][r][s]
                    Let var = (sumSquared[i][r][s]-2*sum[i][r][s]*avg+num[i][r][s]*avg*avg)
                                   / (num[i][r][s]-1)
                    Let t = (|avg|+gamma) / sqrt(var/num[i][r][s])
                    If t > TDistrib(alpha,n-1), (i.e., P(true_avg < -gamma) < alpha)
                         then if avg ≥ 0 then remove range s from the race
                         else remove range r from the race
          If there are not at least two ranges still in the race for weight i
               Set min[i]=min[i][winning r] and max[i]=max[i][winning r]
               If the new width < minWidth
                    then set w[i]=(min[i]+max[i])/2, and fix it there permanently
               Clear statistics for all weights to restart their races
EndWhile
Return w[1..m]
```

Applications of RVSS

RVSS was used to generate **attribute weights** for an **instance-based learning** system so as to be more robust in the presence of irrelevant attributes.

- 4 irrelevant attributes added to each dataset.
- 31 datasets from UCI machine learning databases.

Using RVSS-generated attribute weights yielded higher accuracy in 11 cases, and lower accuracy in only 6, with over 1% higher accuracy on average.

Dataset	kNN	kNN/RVSS
Anneal	90.5	**92.0**
Australian	83.3	83.3
Breast Cancer (Wi)	95.9	95.9
Bridges	**54.3**	50.6
Crx	83.9	83.9
Echocardiogram	93.2	93.2
Flag	59.6	59.6
Glass	**50.5**	49.5
Heart	83.3	83.3
Heart.Cleveland	77.2	77.2
Heart.Hungarian	**82.6**	82.3
Heart.Long Beach	71.0	**72.0**
Heart.More	74.4	74.4
Heart.Swiss	93.5	93.5
Hepatitis	83.9	**85.2**
Horse Colic	67.8	**71.8**
Image Segmentation	86.0	**88.3**
Ionosphere	83.7	**86.0**
Iris	82.0	**90.7**
Led-Creator	**69.2**	68.5
Led+17	68.5	**69.2**
Liver (Bupa)	55.1	55.1
Pima Indians Diabetes	70.4	70.4
Promoters	**84.9**	84.0
Sonar	**79.3**	78.8
Soybean (Large)	85.7	85.7
Vehicle	59.2	60.0
Voting	96.1	**96.3**
Vowel	56.6	**74.2**
Wine	92.2	**93.3**
Zoo	94.4	94.4
Avg	**77.7**	**78.8**

RVSS also applied to generating **weights** in a **multilayer perceptron**.

- It reduced sum-squared error and improved generalization accuracy as the algorithm progressed
- But it was not able to achieve the same level of accuracy as the error backpropagation training algorithm.

Conclusions

• The real-valued schemata search (RVSS) algorithm extends the binary-valued schemata search described by Moore & Lee [1993] to real-valued domains.

• It uses a "blocking race" (BRACE) statistical test to decide when one portion of the search space can be eliminated so as to narrow the search space.

• The work presented here is still in its preliminary stages, but the RVSS algorithm has been applied with some success to weighting of input attributes in an instance-based learning algorithm.

Future work

- Adding the ability to back up and re-widen the search space.
- Direction on what values to use for the parameters:
  - α (the confidence with which decisions are made)
  - γ (the amount of allowable error in a decision), and
  - *minWidth* (the smallest range that is subdivided).
- Careful analysis of how many subranges to use.
- Identification of additional problems that could benefit from this technique, such as finding weights for radial basis function networks.
- Incorporation of the statistical tests into other techniques such as genetic algorithms.