ANNNS SPEC

Design Specification of the Acolyte Artificial Neural Net System.

BY ALLAN SCARFF

CONTENT

INTRODUCTION

Read "AANNS Explained.odt" first!

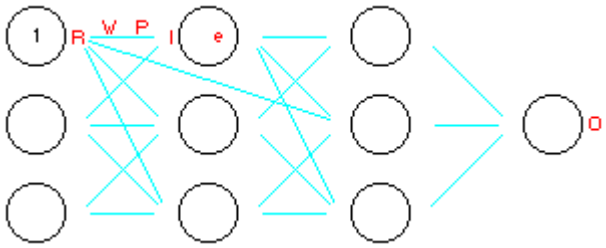
The processing is a hierarchical arrangement that selects a subset of permutations of net structure and net component values with an element of randomness thrown in. Without the randomness, the discovery of useful training advancements would be compromised by the unconscious underlying assumptions built into the processing logic.

The basic hierarchical levels of processing from inner to outer control are :

- Training Ascent - train current active set to produce a trail.
- Training Expedition - control of training ascents to produce a set of trails.
- Contraction - reduce net structure to trial lean unique structures.
- Expansion - when contraction deemed going nowhere expand net structure.
- Pending to Active set migration.
- Reference to pending set migration.

Once the training data has been collected and the schema specified, the above process control is completely automated. The only manual override is to halt the process and use the best net the system can provide. This approach is a practical necessity because the lengthy time spent manually fiddling with net variables is not likely to be productive. If results are disappointing, this means the AANNS analyst should consider gathering more/better training data, and/or splitting the problem into "perception" stages.

The basic AANN structure



[anns001.gif]

The above diagram indicates most (but not all) of the fundamental elements of an AANN. The first column shows three input nodes. Note that no links appear to the left. The last column indicates one output node (although in principle more than one output node could be present). Note that no links appear to the right. The middle columns indicate intermediate nodes. If present, an intermediate node is linked both to the left and to the right. Links direct flow of data from left to right thus this structure guarantees that no loops occur. (Loops and iterations will be introduced later but for now we will consider a straightforward heirarchical net.)

When activated by input from the left, nodes present a zero or positive real value R to each of their right-hand links. The constant node, indicated by "1", is always present. Its job is to present a value of 1 to any right-hand links it owns whenever an input record is activated. The other nodes in the first column are designated to receive input values from each activated input record.

Each link owns a weight value, W, a real value (which can be negative). This is used to multiply the R value to give a product P. The sum of the products of a node's left-hand links provides input I to the node. If the value is less than zero the resulting R value is zero. Otherwise (if I is greater than zero), the I quantity is raised by the power of e. The range of each node exponent e is from 0 upwards. If the value of e is 0 the R value can only be 0 (if I <= 0) or 1 (if I > 0) and so acts as binary logic. If the value of e is 1 the I value is passed through providing an analogue logic (albeit limited by the number of significant bits in the floating point numbers held on the computer). Intermediate values give a compromise and allow evolution between the two types of logic. Values greater than 1 are also possible. The output node R value is rounded to the nearest whole integer O to give the output value from the net.

Link weights and node exponents are normally **unfixed**, meaning they can be updated by the training process. However, especially when creating the initial structures of a net, these components can be **fixed**. This means their value cannot change but the components may still be deleted.

The main design aim is to enable the compact representation of *any* solution in the net structure whilst using the minimum of elements. The temptation to include many different specialised types of node has been resisted because this would lead to complexity and still would not guarantee generality of solution. The elements chosen are a development from a simple Hopfield net. The biggest change is the minimal but necessary introduction of the exponent value to add analogue logic. A second minor innovation is the introduction of the constant node. This allows thresholds to be a fixed zero for every node which simplifies the ANNS generator design. The constant node can be linked to any intermediate or output node in order to fully mimic the effect of a non-zero threshold. A third innovation is that of allowing links across levels of nodes. This is necessary to allow for flexibility in automatic insertion and deletion of nodes and links .

The intermediate data uses real numbers (simulated by floating point numbers on the computer) so that very small trial modifications can be flexibly made during the evolutionary training process. The possibility of negative weights is required for the provision of inhibitory logic. Note that because firing occurs only when the zero threshold is exceeded, the process is not symmetrical according to sign.

The chosen design can cope with negative numbers and imaginary numbers by assigning extra input and/or output nodes. For example, a negative number can be represented by a value node plus a unary node for the sign where a positive sign is indicated by 0 and negative by 1. Where the negative input value is commutative, this has the merit of simplification because the logic concerned with the absolute values can be treated separately from the sign logic, each separately being easier to evolve than if conflated.

The chosen design can cope with fractional numbers by translation. For example, the output node value can represent a whole number of fractions. Internally this is treated as an integer but when the result is displayed for human consumption it is converted to the form specified.

The design as described so far cannot fully cope with iterative logic. This will be covered later on.

An example of an AANN

This section demonstrates the basic ANN structure by showing a potential net solution (the end result of the ANNS generator process) to a simple mathematical problem. Not that you would, but for the sake of simple illustration, suppose you wanted to develop a net to produce the magnitude of the average of any two integers :

x = abs(a + b) / 2.

The terms a and b can be negative so each must be represented by two input nodes, one for the value and one for the sign. In the schematic below i1 holds the magnitude of a, i2 holds the sign of a, i3 holds the sign of b and i4 holds the magnitude of b. The output node o1 gives an integer result, x which has been specified to be automatically divided by 2 to produce the required output. The intermediate nodes (automatically evolved) are m1 thru m6. The exponent of each node is indicated by a superscript character, in these specific cases 0 or 1. The links to the right of each node are indicated with weights inside parenthesis. –g indicates an infinitely large negative value. The schematic is the standard way an ANN is printed out. (The links are not drawn because they would create a mess of cross-cutting lines)

```
i1^1  ( 1)m3
      ( 1)m4
      (-1)m5
                                  m3^1  ( 1)m6
i2^0  ( 1)m1
      (-1)m2          m1^0  (-g)m4
                                  m4^1  (-g)m6          m6^1 ( 1)o1      o1^1 /2
                                        ( 1)o1
i3^0  (-1)m1          m2^0  (-g)m4
      ( 1)m2
                                  m5^1  ( 1)m6
i4^1  (-1)m3
      ( 1)m4
      ( 1)m5
```

Note that the constant node has been automatically removed because, in this solution it was unneeded. Note also that the large negative values have been automatically produced to ensure that the logic will be maintained for very large input values. The following shows an example of data flow with values –7 for a and +4 for b giving x = 1.5 :

```
7i1^17  ( 1) 7m3
        ( 1) 7m4
        (-1)-7m5
                                  3m3^13  ( 1)3m6
1i2^01  ( 1)  1m1
        (-1) -1m2         1m1^01  (-g) -gm4
                                  -gm4^10  (-g)0m6          3m6^13 ( 1) 3o1    3o1^13 /2 1.5
                ( 1)0o1
0i3^00  (-1) 0m1          -1m2^00  (-g) 0m4
        ( 1) 0m2
                                  -3m5^10 ( 1) 0m6
4i4^14  (-1)-4m3
        ( 1)  4m4
        ( 1)  4m5
```

A feature of the above logic, which may not at first be apparent, is a degree of symmetry. i1 and i4 are said to possess symmetry as are the pair i2 and i3. Thus a general solution exhibits certain symmetries in intermediate nodes, links and weights. As will be seen later, by declaring input symmetries, consequential constraints can be imposed on the evolving net to encourage the generality of the solution.

More about node types, topology and linkage.

Intermediate nodes are multipurpose but input and output nodes are assigned types which restricts the ways in which they can connect. **Primary nodes** are implemented by combinations of the more fundamental **secondary nodes**. Primary node types are defined by the combination of content type and topology type.

Content type can be one of the following :

- **unsigned real number** contains any positive real number.
- **unsigned integer** contains any positive integer.
- **signed real number** number automatically mapped onto unsigned real number and binary node for sign.
- **signed integer** number automatically mapped onto unsigned integer and binary node for sign.
- **indicator** positive integer automatically mapped onto one or more binary node elements.
- **identifier** hashed string mapped to significant volume binary node elements only.

Indicator nodes are merely a convenient way of specifying a number of binary nodes. It is important to use indicator nodes to prevent the system spuriously creating logic based on magnitude where that is irrelevant. Typically an indicator will only have values that produce one bit, i.e. 1, 2, 4 , 8 etcetera. However, it is possible to use more than one bit e.g. the value 5 would set two bits, 1 and 4. An indicator node is the correct type to use for indicating infinity (+ or – or neither) rather than attempt to use very large numbers as an approximation.

Identifier nodes use unique strings. If the internal set contains more than twelve records with the same string, this is allocated a binary node, otherwise the content is ignored for training purposes. The reason for using identifier and indicator nodes instead of separate binary nodes is that, at certain stages, the identifier or indicator node is treated as a logical entity, not just as a collection of unrelated binary nodes. The identifier type is only applicable to input nodes.

Each input binary element is augmented by an intermediate node to represent the negated value. The intermediate node is allocated at initialisation with a frozen superior link weight and a frozen exponent.

The full list of topology types it is intended that AANNS covers at a first implementation is :

singleton	single entity - not an array
heap	array with no spatial organisation, each element independent of the others
line	one dimensional array with optional left/right symmetry
square grid	two dimensional square array with up to eight-way symmetry.
rectangular grid	two dimensional rectangular array with up to four-way symmetry

Heaps, lines and square grids are all examples of **arrays** that match output elements to input elements. A heap is distinguished from a line and a square grid by the property of each element being independent of any spacial position. Each element in a line is adjacent to two others except at the ends of the line where they are adjacent to only one element. A grid element is adjacent to upto four orthogonal and four diagonal elements. A square grid, a rectangular grid, and a line element, can be linked to each adjacent element in a special way in order to transmit patterns based on symmetry.

Secondary nodes for holding coordinates of arrays are automatically allocated :

- **x-coordinate** for line, square grid, and rectangular grid but not heap or singleton.
- **y-coordinate** for square grid and rectangular grid but not heap or singleton.

For a line array with no symmetry, the x-coordinate is automatically supplied with the **physical offset** n.

For a line array specified as having left-right symmetry, the x-coordinate is supplied with the calculation of the **logical offset** :

x-coordinate = w/2 - |(w/2-n)| where w is the number of elements in the line array.

Note that | | denotes absolute value (always positive dropping the minus sign).

For rectangular grids left-right, top-bottom, and the combination of the left-right and top-bottom symmetries are possible (as well as no symmetry). For square grids a further four combinations can be formed by the addition of rotation by 90 degrees. The x-coordinate and y-coordinate are calculated using the table below where w = number of columns and d = number of rows in the array and x and y are the array coordinate offset values input :

Symmetry	**x-coordinate = ?**	**y-coordinate = ?**
None	x	y
left-right	w/2 - |(w/2-x)|	y
up-down	x	d/2 - |(d/2 - y)|
left-right + up-down	w/2 - |(w/2-x)|	d/2 - |(d/2 - y)|
rotate	y	x
rotate + left-right	d/2 - |(d/2 - y)|	x
rotate + up-down	y	w/2 - |(w/2-x)|
rotate + left-down + up-down	d/2 - |(d/2 - y)|	w/2 - |(w/2-x)|

It is also possible to "twin" input nodes of the same type by declaring them as being logically symmetrical. This is then taken into account when structural alterations are made to the net. Any subordinate link added to one node must also be added to the other and an identical weight must be maintained. Subsequent deletion of one of the paired links must result in deletion of the other. More generally, any structure and component values of intermediate nodes and links connected to the twin input nodes must be maintained as carbon copies at all times.

Two quantities are generated for both input and output nodes. These quantities can be used in initialising link weights, calculating “reverse” secondary output node content, and in the calculation of the magnitude of secondary input nodes but not for adjacent holders :

- **min** supervisor defined minumum value, defaulted zero (unsigned) otherwise very large arbitrary negative number.
- **max** supervisor defined maximum value , defaulted to an arbitrary very large number.
- **range** **max** minus **min**.
- **maxabs** the greater of the absolute of **min** and **max**.

For arrays the same single min and max quantity is used for all elements. The default overrides any supervisor defined min or max quantity which is found to be contradicted by a data value.

For primary input nodes, secondary node types are generated internally to complement real number, integer, indicator and array types :

- **sign** binary node containing zero or one for +ve or -ve with, for input, a fixed zero exponent.
- **rev-sign** binary node containing negated sign again with a fixed zero exponent.
- **binary** binary node containing value for indicator element with, for input, a fixed zero exponent.
- **rev-binary** node containing negated binary value for above.
- **mag** magnitude of real/integer number above minimum.
- **rev-mag** replacing mag where this simplifies structure, **range** minus **mag**.
- **abs** absolute value part of signed (or unsigned) real/integer number.
- **rev-abs** replacing abs, where this simplifies structure, **maxabs** minus **abs**.
- **array-size** the number of elements in a heap or line or grid.

The array size is only used as input when the array is **variable**, i.e the size of the array differs from record to record. For **fixed arrays**, the content of an array size is always the same and is therefore of no direct interest in generating a specific net. However, if at a future time, additional records containing variable sized arrays are introduced, the existing array sizes become active.

For any one net, arrays can be of variable sizes from record to record but only one size is allowed across more than one array in any one record. The **array-si**ze, thus applies to the individual record rather than the array node. Input nodes are then automatically linked to output nodes mapping singleton to array, array to singleton, or array element to array element. Logically, multiple declared arrays are treated as a single array with the aggregation of all the elements.

Primary output nodes also generate rev variants for use as input in iterative processing (see next section).

Secondary binary nodes require each indicator element to be linked separately. For example, an indicator with a maximum value of seven would be mapped onto three elements separately linked. On the other hand, arrays elements have only one structural link which is shared between all the elements of the array. Changing the link weight affects all

the elements of an array. Indicators can be combined with array types. In this case the links for the indicator are each shared between each element o the array.

The negation variants of binary and sign secondary nodes are included to simplify initialisation of links and to provide logical equality between the binary values when forming structures.

The following types of secondary node types are potentially created for primary output nodes:

- **sign** for signed numerical output (fixed zero exponent).
- **binary** binary node containing value for indicator element (fixed zero exponent).
- **abs** absolute value part of signed (or unsigned) real/integer number.

A training record containing array information is split into element instances with the relevant secondary node content. In a record instance, the coordinate information is specific to that element, whereas the array size and singleton nodes share the same data across all elements. The mapping logically restricts the linking between node types. A grid output node is normally connected (directly or via intermediate nodes) to a grid input node of the same dimensions, but it can also be additionally connected to singletons, each element taking the same linkage input. Although rare it is even possible for a grid output node to be connected solely to one or more singletons, the difference in element values relying solely on x-coordinate and y-coordinate input. Similarily, a line output node is normally connected to an line input node but can be connected to a singleton and can rely on value differentiation solely on x-coordinate input. Again, a heap output node is normally connected to a heap input node of the same size but can be connected to a singleton input. Additionally it can also be connected to line or grid input nodes if the number of elements is the same, in which case the x-coordinate and y-coordinate information is suppressed.

For arrays iterative processing can take place between adjacent (orthogonal or diagonal) elements. To this end a number of input and output **adj** nodes are allocated :

- **adj-N** square or rectangular grid
- **adj-NE** square or rectangular grid
- **adj-E** square or rectangular grid or line
- **adj-SE** square or rectangular grid
- **adj-S** square or rectangular grid
- **adj-SW** square or rectangular grid
- **adj-W** square or rectangular grid or line
- **adj-NW** square or rectangular grid

These are logically connected as follows (links between nodes across rows) :

Superior adjacent element output node	**Input node**	**Output node**	**Subordinate adjacent element input node**
S element adj-N	adj-S	adj-N	N element adj-S
SW element adj-NE	adj-SW	adj-NE	NE element adj-SW
W elemnt adj-E	adj-W	adj-E	E element adj-W
NW element adj-SE	adj-NW	adj-SE	SE element adj-NW
N element adj-S	adj-N	adj-S	S element adj-N
NE element adj-SW	adj-NE	adj-SW	SW element adj-NE
E element adj-W	adj-E	adj-W	W element adj-E
SE element adj-NW	adj-SE	adj-NW	NW element adj-SE

The subject element's output adj nodes are initially connected to all input nodes except the array generated nodes, x-coordinate, y-coordinate and array-size. The subjects input adj nodes are initially connected to all output nodes via zero weighted but fixed links (counting towards the link total only when their weight is other than zero).

An input/output node can optionally be further qualified by one or more incidences of semantic type. Each semantic type is a unique label which limits the usage of the AANN when being considered for derived input (see later). For example, racecourse would be a suitable semantic type for an indicator which identifies specific racecourses. In this case it is very unlikely that the AANN could sensibly use any other semantic type of indicator in its place. Note that the same semantic type can be applied to different topological node types.

In order to facilitate training a **memory contributor** intermediate node is introduced for “remembering” a single value output from a unique set of inputs. By effectively removing the record instance for the selected output node and load set

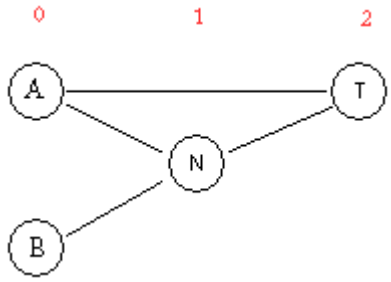
record, training which may have been in conflict is freed up for other records. Memory contributor nodes can be readsorbed at later stages of training.

Loops and iterations

One might suspect that in any organic net system, links will inevitably be made that break the one direction rule and thus create loops. In the general case, creating new **backwards links**, when expressed, tends to create either zero or very large node output values. In training, an organic net would prune such links because the output would be unusable. Very occasionally a backwards link can create convergent node output values, i.e. after each iteration of the loop an output value gets closer to a stable value. This ability to express iterative logic is the most valuable addition to acquire.

How exactly the human neural net organises loops is an area for further investigation. However, there are a few speculations worth mentioning. Although the number of iterations of any one loop must effectively be limited in order to present an answer in real time, it is difficult to contemplate a simple mechanism that would produce an exact number of iterations suited precisely to the desired collusion. If we assume that a human neural net is limited to an indefinite number of iterations, this limits the trained logic to convergent node output values. The ANNs implementation takes this simple approach, but with some twiddles in order to improve efficiency given that the training is executed on a fast but ploddingly serial computer. [An alternative scheme is to introduce a decay factor to force otherwise divergent behaviour into convergent. However, this foregoes the pruning that, without the decay factor, occurs naturally, and might overall be counter-productive.]

In order to more easily understand the implementation, concepts such as “node levels” and “net layers” are introduced. These concepts are not fundamental to the design – they merely make the design easier to conceptualise.

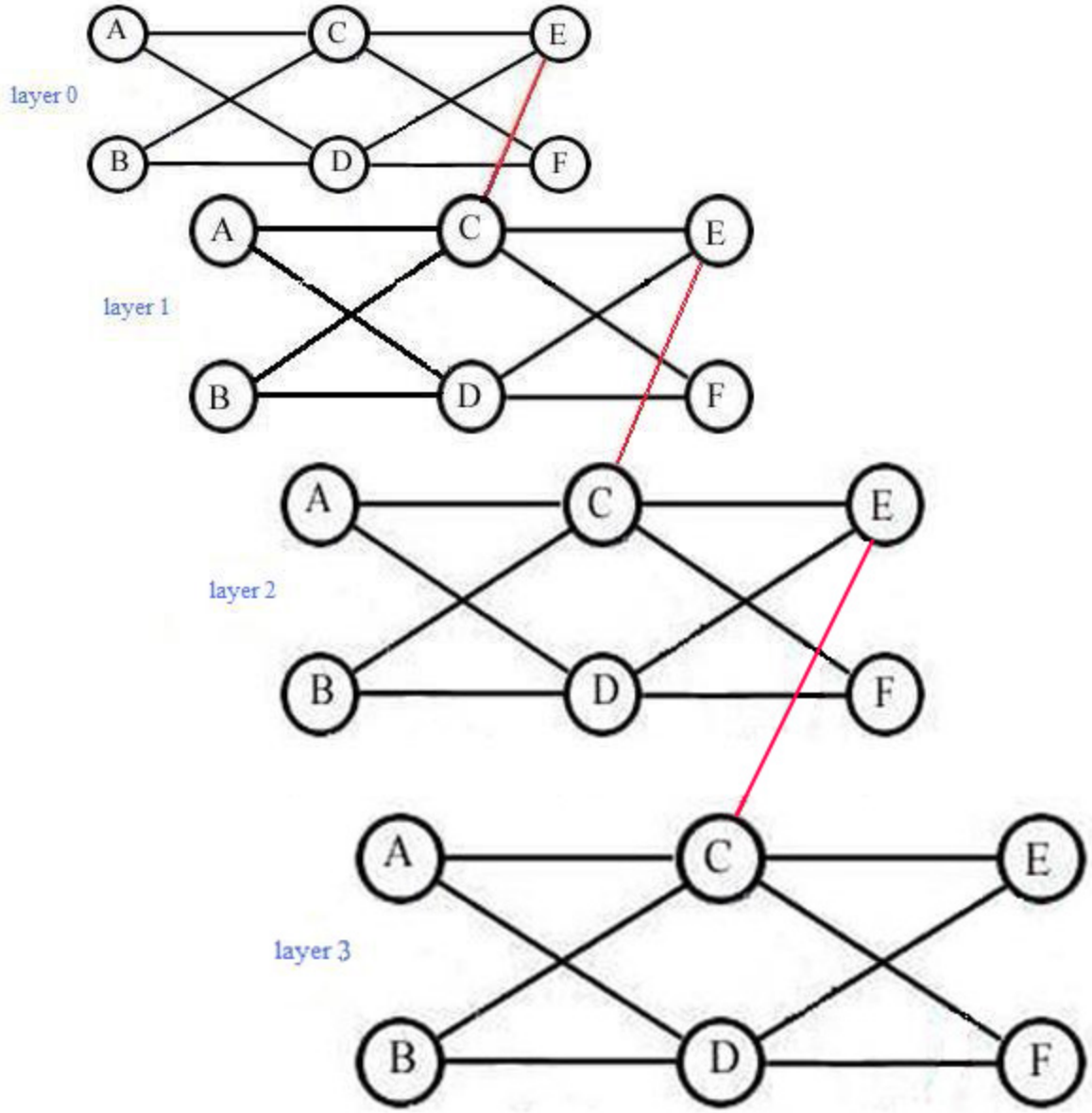


[anns039.gif]

In the above diagram three hierarchical **node levels** (0 1 2) in a simple net are shown. These levels are determined by the number of propagation steps needed to express each node. Although the output from link AT is available after one step, the NT output, needed to complete the total input to node T, is not available until the second step. Hence node T is at the second level of the net. Note that adding or subtracting links to a net can significantly change the hierarchical levels owing to changes in propagation.

If a link is added where the subordinate node is not at an existing subordinate level, it may cause increases in subsequent node levels, or cause a propagation loop. If it does the latter, the link is *notionally* disallowed. Instead, *conceptually*, an extra **net layer** is created. This has the same structure with the same input node values as in the existing layer. The **backwards link** is then notionally connected to the node in the newly created layer. The input value for the backward link is taken from the superior node in the existing layer. This is used in expression of the new layer, which thus avoids a loop.

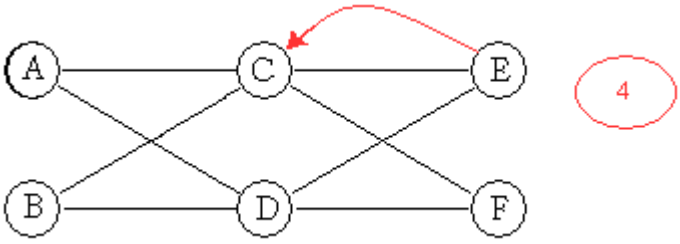
[anns040.gif]



The above diagram shows how to regard the backwards link EC. For level 1 the E node is treated as a superior node at the previous level, using the node E output value from the expressed layer 0. In general, each layer, in turn, is expressed level by level in turn. The node output quantities for the previous layer are retained for use in expressing the next. For layer 1 and beyond, if no new input to a node is available, then the previous layer result for that node is taken (rather than work it out again). If there is no change for the layer from the previous set of results, both for output and intermediate node output quantities, then no further layers need to be processed. This truncation can be also be applied when detecting *any* repetition of results. The detection can be implemented by forming a checksum for each level based on a hash algorithm applied to each node output quantity.

A more compressed representation of the previous four layer net is shown thus :

[anns037.gif]



The red 4 in an elipse indicates the **net iteration count** (conceptually the number of net layers). The red line pointing from node E to node C indicates the backwards link.

Normally a completely trained net, if containing backwards links, will have a high net iteration count in order to give as accurate an answer as possible. However, during training, for much of the time, nets will use much lower net iteration counts for efficiency reasons.

The adj node connections are not limited by a net iteration count, instead limiting the number of iterations to just reach the the edge of the grid.

Back-propagation is always applied to the "expanded" net after iteration has been logically applied.

Introduction to ANNS sets

The totality of all possible records representing all possible values and combinations of values both input and output is referred to as the **universal set**. Usually the training data is a small but representative subset of the universal set and is referred to as the **internal set**, and the remaining records are referred to as the **external set**. The ANNS first reserves a portion of the internal set to form a **reference set.** In contrast with most conventional systems this is achieved automatically. The reference set is used later to test how general the current applicability of the trained net is. It is important that the reference records are never used for training, otherwise their validity as a "gold standard" would be compromised. The remaining records in the internal set form the **load set**.

The first large departure from conventional systems is to use, at any one time, only a few records for active training – thus forming a subset of the load set termed the **active set**. The first advantage this gives is that a training cycle can be completed more quickly i.e. in proportion to the ratio of the number of selected active records to the whole. You might think that the loss of representational diversity would be correspondingly limiting. However, as the training proceeds, new selections of records to be included in the set are applied to the current developed net which thus retains, by inertia, some of the ability to satisfy records no longer included in the active set.

The second advantage is that the training process, since it is fitted to each subset of records along the way, more easily finds partial solutions which would otherwise be suppressed in any attempt to satisfy *all* records. These partial solutions may be totally superseded at a later stage, but may form an essential step in the path towards the ascent of Mount Improbable.

The third advantage is that a staged process guards against over-training. Once a net gives accurate results for the training data, using further conventional training without new data is pointless because no further corrective action is available. Thus if the solution found is not of good quality as regards its general applicability, it is difficult to remedy the situation. One method would be to start from scratch and trust to chance that a better result will ensue, but this is an inefficient process. In contrast, by subsetting the records, each stage is, in part, an efficient check that the previous training is generally valid, and, if not, encourages automatic correction. The representational quality of the net is built up in stages, and since the full set of training records is not precluded, it is statistically far more likely to be better than a single stage using all the training records. The much simpler traditional method of repeated application of single records by a somewhat arbitrary number of iterations does have some of the same effect. However the new method of inclusion of new records and replacement of old in the active set, allows a much more sophisticated control to apply the requisite amount of training at each stage. The efficacy of each training stage can be tested by trying out the entire load set. If the overall load set score is significantly degraded, in subsequent training stages the net can be encouraged towards its previous better state.

The size and composition of the active set should ideally use the minimum number of records with a sufficient compliance to targets so as to stabilise the net such that advancement of the active set is also likely to advance the complete load set. On the other hand, sufficient target discrepancy is needed to "challenge" the net into evolving upwards. In order to facilitate finding the best size for the active set at any one moment, the set is further divided into a **current active set** and a **recent active set**. The current active set holds the records that are used for immediate training. Some of these records can at some later stage be placed in the recent active set so that training can continue with a smaller set of records. This can extend the training life of the active set by allowing the side-tracked records to recover and thus reduce the overall score of the entire active set whilst still improving the pending set score. It has secondary aim of allowing progress by side-tracking bad data.

The records in the load set that do not form the active set, form the **pending set.** This set is used to judge the efficacy of training the active set as well as providing records to expand the active set when needed. It thus acts a pseudo reference set. This is necessary because the reference set would be compromised if used to control the training. The accuracy of the pending set is compromised but it is still sufficient for determining the likely improvement of the net.

The ANNS back-propagation method

This section gives some of the detail of the method used to train a neural net at an innermost level, i.e. when applying a training record in order to prepare for the selection of an individual component update.

First the training record is *expressed*, i.e. the input quantities are processed through the net structure to give intermediate quantities and output node results. The back-propagation process is then propagated in the exact reverse order starting at the output nodes with the supplied output node target range quantities, and working backwards via owning (i.e. superior) links to propagate intermediate target quantities for each and every net component. Each intermediate target quantity is individually calculated to achieve the whole of an output node target or failing that as near to the whole output node target quantity for output from the component to the next element of the net structure. Note that where target ranges are supplied for output nodes, which does not match the expressed quantity the intermediate target is calculated to meet the closest bound. "Don't care" output nodes, i.e. output nodes with null content, own the maximum target range so that all intermediate targets are deemed already matched.

The propagation for each net component produces an **output target range**, and from that an **input target range**. Each **target range** consists of a "centre" **ideal target range** or **ideal target value**, a "surrounding" **extended target range** (lower and upper parts), and "surrounding" that, a **tolerance target range** (again lower and upper parts) . The lower and/or upper bounds of each range can coincide with the adjacent range to reflect a "null" range. The ideal target range is that range of contiguous quantities that produce a perfect score. If only a single value represents the perfect score, then this is more corrrectly referred to as an ideal target *value*. The extended target range is, if present, the range of quantities below and above the ideal target range such that above the lower bound and below the upper bound it gives less than perfect scores above zero. Thus increasing a quantity within the lower extended target range potentially improves the score, and decreasing it potentially reduces the score. For quantities within the upper extended target range the exact reverse is true. The tolerance target range is those quantities below and above the extended target range, i.e. those (contiguous) quantities which score zero. Null ranges can occur, but if present the extended target range must surround either an ideal target range or ideal target value although either the lower or upper part can be null corresponding to a minimum or maximum ideal target range/value. It is possible for there to be null ideal target and extended target ranges leaving just a maximum scope tolerance target range.

The detail logic is best understood from considering specific examples:

First consider the output node 4o1^{2}16{4…9}[0…256]
where 4 is a node input quantity
where o1 is an output node identifier
where ^{2} is an exponent value
where 16 is the calculated **node output quantity**
where {4…8} is a data specified **ideal node output target range**, i.e. any quantity between 4 and 9 inclusive is deemed completely correct.
where [0…256] are the minimum and maximum acceptable quantities for output (>256 is treated as = 256).

An **extended node output target range** for o1 is calculated to include all values which would result in an improvement on the current score, in this case greater than zero and less than 256 and not 4 to 9. Combined with the output target range this is shown as : {0…4{4…9}9…256}.

A **tolerance node output target range** is calculated to include all values which would result in no reduction on the current score. The combined node output ranges are shown as : {0{0…4{4…9}9…256}g}. g stands for a system defined very large number (but set low enough to avoid overflow problems).

The corresponding combined **node input target ranges** are next propagated via the node output target range, giving {0{0…2{2…3}3…16}16…g}. The entire node targets are shown as 4{0{0…2{2…3}3..16}16…g}o1^{2}16{0{0…4{4…9}9…256}256…g} [0…256]

The **link output target range** is determined by the subordinate node input target range and the subordinate node input quantity. Consider the extension to the example by a superior link m2 2 (-1) –2 o1 where –1 is the weight, 2 is the intermediate node, m2, node output quantity, and –2 is the the **link output quantity**. Since the o1 node input quantity is 4, this means that a quantity of 6 is presented from other superior links. Taking this into account we calculate this link's output target range to be {-g…-6{-6…-4{-4…-3}-3…6}6}. The **link input target range** is {0{0…3{3…4}4…6}6…g}. Because of the negative weight, the order of the quantities specifying the ranges are naturally the reverse of the output, but also negative quantities are "truncated" to zero because nodes can only output positive quantities.

If the superior node owns only the one subordinate link, then the superior node output target range equates to the link input target range. However if more than one subordinate link exists, the link input target ranges must be "merged" to

form the node output target range. For purposes of training and expansion, instead of the logical splintering of ranges, a single “simplified” merged range is produced. This range can either be **coherent** or **incoherent**. A coherent range is produced when both, all the superior ranges considered are coherent, and also there is an intersecting common ideal value or range. Otherwise an incoherent range is produced.

To construct a coherent node output target range, first the highest lower bound and the lowest upper bound of the ideal link input target ranges of each of the subordinate links is taken. If the resulting ideal range has a lower bound equal or less than the higher bound, then the *lowest* lower bound and the *highest* upper bound of the superior links extended input target ranges are taken to form the merged extended range. The same process applied to the tolerance range limits is used to form the merged tolerance range. For example the subordinate link input target ranges shown :

{0…6{6…9{9…12}12…14}14…g}
{0…5{5…10{10…15}15…100}100…g}
{0…2{2…8{8…16}16…140}140…g}

form

{0…2{2…10{10…12}12…140}140…g}

For an incoherent range the ideal range is instead formed from the lowest lower bound and the highest upper bound of the superior links input ideal target ranges. For example :

{0…6{6…9{9…12}12…14}14…g}
{0…5{5…10{10…15}15…100}100…g}
{0…2{2…13{13…16}16…140}100…g}

form

{0…2{2…9{9:::16}16…140}140…g}

The colons denote an incoherent range, here because there is a gap between the 9…12 and 13…16 ranges. This incoherent merged range can then go on to propagate futher superior incoherent ranges.

The significance of coherent versus incoherent ranges, is that wheras components with quantities within either coherent or incoherent ranges are passed over for candidature for update in the training accent process, incoherent ranges lead to splitting of links to produce coherent ranges in the expansion phase.

Binary nodes are propagated in the same way as non-binary. The ideal target for an output binary node can only be 1 or zero. This translates into node input target ranges of {0…1{{1…g}}} and {{{0}}0…g}. Note the null extended ranges. Merging the previous two contradictory ranges gives the incoherent range {{{0:::g}}}. Note that this indicates that regardless of existing input, no change is anticipated to improve the score.

By propagating backwards some components are identified to which improvements might be profitably made, i.e. changes would improve the score, and the rest of the components are ignored since changes might lower the score. Note that complex net structures might lead to components which are eschewed despite the possibility of improvement. This is because we seek improvements over a range rather than for a specific instance which is more likely to prove arbitrary.

To qualify for potential amendment, a component must be **active** and also have an input quantity outside its ideal target range.The difference between the input quantity and the neartest target quantity is called the **target update quantity**. If the component has null ideal and extended target ranges but does possess a tolerance range, then it is classed as **inactive**, otherwise **active**. To qualify for potential amendment, a component must be **active** and also have an input quantity outside its ideal target range.

Where backwards links are included, the expression and back-propagation can be understood by considering the conceptual net layers. For expression, starting with layer 0, each layer is expressed in turn. For back-propagation the reverse is true. For the first implementation, only the target quantities of layer 0 are used in the contraction and expansion processing. The targets in the added layer are purely intermediate.

Overview of the ANNS process control

The basic hierarchical levels of processing from inner to outer control are :

- Training Ascent - train current active set to produce a trail.
- Training Expedition - control of training ascents to produce a set of trails.
- Contraction - reduce net structure to trial lean unique structures.
- Expansion - when contraction deemed going nowhere expand net structure.
- Pending to Active set migration.
- Reference to pending set migration.

Overall the heirarchical control routines maintain a number of nets along with their corresponding sets of training records and scores. One **optimal net** (scored from the pending set) is kept for each unique number of links unless superseded by the same or a better score at a lower number of links. Additionally, as they are formed, each **optimal net** is examined to see if it qualifies as a new **best net**.

A somewhat hidden design principle is employed in determining the detail control method. A number of different factors are involved in training the data records. Because the numbers are too great, no one factor is exhaustively explored. Instead each factor is lightly tested in one context but repeatedly testing in varying contexts. This serial process compensates for lack of parallel processing power by allowing the more promising combinations of factors to emerge quickly.

The first and innermost level of control is concerned with efficient training using the current active set and the current net structure, developed from a **base** point on the solution attribute terrain, input from the next level of control. The main principle used is to find alterations to components that improve the score of more than one active set record with the anticipation that a number of such small alterations will be statistically more likely to improve the pending set score (of course avoiding the need to expensively test each change against the relatively large pending set). The aim expressed in topographic terms is to efficiently ascend the local slope from the base and, despite minor dips, explore any plateaux or ridges to find the **local peak**. The local **training ascent** cannot be guaranteed to find the highest local point but a degree of variability allows the next level of control to repeat the same local ascent with a potential different outcome.

The second level of control is concerned with selecting the starting points for performing a series of training ascents operating on a fixed set of nodes and links (but varying the component values). The string of ascents is termed a **training expedition**. The initial starting point for the first training ascent is determined by a **bedding pass**. Thereafter, the history of training ascent trails is used in order to determine potentially good points from which to start further training ascents. (This complex process is described in the next section.)

The next level of control is the structural contraction of nets selected from the set of optimal nets. Any repeat of any previously trained structure is prohibited. The number of candidates is further reduced by a calculation as to the likelyhood of success based on previous history. Each selected modified net is submitted to a training expedition. If any contraction was successful this results in new optimal nets which will most likely throw up further cadidates for contraction. When all the candidates have been exhausted the contraction phase is ended and the expansion phase begins.

The structural expansion of nets is again selected from the set of optimal nets. Again any repeat of any previously trained structure is prohibited. Again the number of candidates is further reduced by a calculation as to the likelyhood of success based on previous history. Each selected modified net is submitted to a training expedition. If any expansion was successful this results in new optimal nets which will most likely throw up further cadidates for expansion. When all the candidates have been exhausted the expansion phase is ended.

The next level of control is concerned with selecting a set of records to form a new **current active set**. The collection of training expeditions for each new current active set is termed an **active training cycle**.

The selection is first based on determining the required size of the current active set. If the previous active training cycle failed to produce any new optimal working net, then the desired size is increased by just one record. If the previous active training cycle produced a net with a superior score to all previous optimal working nets, then the desired size is reduced by one record. Otherwise the desired size remains the same.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

The next level of control is concerned with selecting the records to form the **current active sets** from which potentially a series of training treks are undertaken. The whole active set is chosen for the initial training expedition. If this expedition is unsuccessful, i.e. does not result in an improved optimal net, then control is ceded to the next level. But, if successful, a number of training expeditions (determined by a binary chop method) are potentially undertaken in order to determine the smallest current active set that can produce both a successful result and a *lowest* active set score. The purpose of this is to eke out the training capacity of the active set since, once trained to a 100% score, no further progress can be made without transferring records from the limited resource of the pending set. Those records which used to belong to the current active set are sidelined to the recent active set – they are not directly returned to the pending set lest they effect the ability of that set to measure progress. The collection of training expeditions undertaken with the different selected current active subsets upon the same net structure is termed a **training pass**.
[More work needed – continuity of net, expanding the current active set when failing to produce optimal net, detecting difficult records to be retained in the recent set]

If the training pass was successful a new training pass is embarked upon and this is repeated (part of the fifth level of control) until an unsuccessful training pass occurs upon which the **active training phase** is completed and control is ceded to the sixth level. But if the first training pass failed to find an improved optimal net then potentially the balance between active and pending sets is altered (the alternative part of the fifth level of control). Evolution of the net is fastest when the active set is large enough to be representative but also the pending set is large enough to check that the improvement in active scores is of general impact. However it is difficult to predict in advance the ideal split. And once the size of the active set is selected there is no going back even if it later it appears to be larger than the optimal size. So a strategy is needed to allocate as few new active records as possible. Thus the pass is repeated on a trial basis allocating records using a binary chop. Essentially, if unsuccessful, half the remaining target pending records are transferred for a new pass, but, if successful, the target number is reset to that current, the number to be transferred is reduced to halfway between the current number and the previous unsuccessful total, and the transfer is operated on the original unsuccessful net. The fact that a number of passes might be necessary to find the minimum number of transfers to become successful is commensurate with the sought for progress. The qualifying improvement is that the total score for the records in the new pending set must exceed the total score for the same records by an amount formulated to reflect the higher error from a reduced number of pending records, plus if applicable, the increase in the volatility in scores between the new and old pending sets. If no improvement is forthcoming, then the trek phase is considered completed and control is ceded to the fifth level of control without replacing the existing optimal net but setting its contraction flag to prevent an attempt of net structure contraction.

The sixth level of control is concerned with expanding and contracting the structure of the net. Each optimal net has two attributes, an expansion flag set to indicate that expansion has been attempted, and a contraction flag to indicate that contraction has been attempted. When a new net is created the expansion and contraction flags are cleared (except for a minimal net which, because it has the minimum structure, has its contraction flag set). At the completion of training of a specific structure net during which a new optimal net may or may not be produced, a search is made of the all the optimal nets including those created during previous training processing, starting from the smallest until one with a clear flag, contraction or expansion, is found. If a clear contraction flag is found, the flag is set and a **contraction phase** is undertaken to create, subject to qualifying as optimal, a new (fully trained) net. If instead a clear expansion flag is found, the flag is set and an **expansion phase** is undertaken to create, subject to qualifying as optimal, a new (fully trained) net, its expansion flag is set. If no optimal net with a clear flag is found then the evolution iteration is deemed over and control is ceded to the outermost level control loop.

The seventh and outermost level of control is concerned with the balance of training records between the reference and the load sets. The training process normally commences with randomly splitting the training records into equal sized reference and load sets. This set population is fixed for the duration of the **evolution iteration**. The iteration is terminated when the net is well and truly “stuck” and cannot be further evolved with the current load set. Then, if the reference set is not 100% matching, a number of records are transferred from the reference to the load set and the training process can then be tried again. Assuming that the problem is a lack of sufficient training records, after a few evolution iterations, this will eventually result in a net with the maximum score despite the increasing error margin as the size of the reference set decreases. However, if the reference set is found to match 100% then the evolution iteration is repeated with fewer load records using the net with even fewer load records as a starting point. Thus potentially a best net might be produced with 100% reference set match but with more reference records a smaller error margin.

The ANNS training expedition

As mentioned in the previous section, the **ANNS training expedition** refers to the set of training ascents operating on a fixed net structure with a fixed active current set. The aim is to explore the solution attribute topology in as statistically efficient a way as possible whilst simultaneously minimising the risk of missing an isolated peak.

The analogy, in 3D terms, is to repeatedly parachuting into a landscape cloaked in dense cloud and, at each attempt, to make an upwards trail terminating at a local peak. The steps of previous trails are marked with beacons that transmit coordinates including height. This information allows for a potentially more efficient, albeit more complicated strategy than that of just random jumps. There are four major aims:

1. Exclusion of areas close to the existing trails to prevent unnecessary duplication of effort.

2. Concentration on the unmapped areas between local peaks. This is based on the statistical likelihood that the highest peak is surrounded by lower peaks, there being trends of elevation in any landscape. It is also based on the possibility of losing the upward path on an ascent, thus giving the possibility of picking up the continuation at a small distance from the break. And finally it is based on the statistical likelihood of the presence of ridges in the landscape.

3. The gaps between projected mountains must be covered so as to detect new mountains.

4. Except for in between peaks, the area of exploration should be limited to areas adjacent to those mapped so as to avoid large number of “ascents” from the midst of large lowland expanses.

Given massive parallel processing capability, designing an algorithm to optimise between the four aims would not be so difficult. A cellular automaton could easily assign a value to each cell in the solution attribute landscape. However, because the actual topology involves as many dimensions as components, this would require a prodigious total amount of computation even though the successful path length might be quite short. Given only the serial processing capability of a typical PC, trying to simulate a cellular automaton would, on average, give rise to elapsed times that reflect the very high total amount of computation. So in practise, a more efficient scheme is required. Because this naturally leads to a much more complicated algorithm, a compromise between efficiency and complexity must be struck.

The first simplification is, when determining distances between any two points in the landscape, to eschew multi-dimensional geometrical calculations in favour of simply adding up the absolute differences using units of training steps. (The allocation and maintenance of training step sizes for each net component will be dealt with later.)

The second simplification is to lose the bulk of points in the trail and use only the base and peak points. The working assumption is that of a fairly straight trail joining base and peak points. In the case of torturous trails, this leads to the possibility of selecting a trail base on or near an existing trail, but revisiting complex topology is not so bad because it might lead to the discovery of a new “hidden” peak. By encouraging selection of points away from both base and peak points, the intermediate trail points can usually be avoided. Typically a mountain will be discernible by a single peak surrounded by base points.

The third simplification is to idealise the sought for distance of the new base point from adjacent base and peak points. For an adjacent peak, the ideal distance is deemed to be equal to the greater of three training steps or the average distance (in training step units) to that peak’s existing base points. For an adjacent base point, the ideal distance is deemed to be equal to the greater of three training steps and the distance between the base point and its peak point. The valuation of how badly placed a potential base point is, is given by summing individual detriment values for distances to each other point in the landscape. Points which are further away than their ideal distance give zero values. Points that are coincident with an existing base or peak are given an arbitrary high detriment value (avoiding overflow in the following calculation). Otherwise the detriment value is calculated by taking the square of the ratio of the ideal distance divided by the current distance and then subtracting one.

The fourth simplification is to define **peak bridge points** so as to facilitate seeding between peaks. A peak bridge point is one located on a line joining a pair of peaks such that it bisects two points on that line (or extended line), each point being at the appropriate peak ideal distance from its peak. Furthermore, to (newly) qualify, the peak bridge point must either be closer to each peak than any base point, or be further away from each base point than each base point’s peak ideal distance. Finally a point cannot (newly) qualify as a peak bridge point if it is less than three training steps from any peak point.

Before attempting the first training ascent for a specific net structure, there must first be assigned an initial base point and peak. The initial peak is assigned simply to the current set of component values. The initial candidate base point is assigned to a set of component values calculated by the following procedure:

A record is chosen at random from those records in the active set which have an imperfect score. The back propagation method is applied to the chosen record. The update quantity for each active component is divided by the number of active components and each result is applied to produce a candidate peak. If the candidate peak is less than three training steps away from the candidate base point, then the candidate base point is extended along the line of connection between it and the candidate peak, so that it is equal to three training steps away. The pending set scores are calculated for both candidate peak and base point configurations and, in the unlikely event of the candidate base point pending score being higher, the base point and peak are swapped and the usual best net update procedures are performed. The candidates are then confirmed.

For subsequent training ascents a new base point is chosen by the following procedure:

A list of peak bridge points is accessed and, if any are present, one is chosen at random to be the new base point. If no existing peak bridge point is found, an existing base point is chosen at random and then an iterative random walk is performed until either no improvement towards a zero detriment score can be made, or, the distance from the selected existing base point exceeds its ideal distance. The resulting walk locates the new base position at a suitable distance from other points without being too remote. The detail process for each step is as follows:

A component is chosen at random from those components with a non-zero training step size. A single training step is applied away from the selected existing base point. The detriment scores for all existing base points and peaks are summed. If the sum has not increased, the update is accepted and the next step is taken unless the current distance from the selected existing base point is greater than the ideal distance, in which case the new base point is deemed found. Otherwise the step is deleted and a new component is chosen at random. This process is repeated until either a qualifying step is detected or the choice of components is exhausted. In this latter case the current configuration is taken to be the new base point.

Having selected a new base position a training ascent is undertaken. This may or may not result in a new peak and may result in fewer peaks if merging occurs. If a new peak is produced and is coincident with the base point then this point is treated, as an isolated candidate peak without a base point (which may later gain base points or become merged with another peak). If a new peak point is produced or the ascent process merges peaks, then the list of peak bridge points must be updated :

Existing peak bridge points which are dependent on erased peaks are first erased. Then each peak bridge point is examined to see if it is closer to the new base point than it is to either of its pair of peaks, and if so the item is erased from the list. For a new peak each pairing with each existing peak is examined. The coordinates of the intermediate peak bridge points are calculated as follows:

For each of the pair of peaks, a point is calculated along the line joining the peaks a distance equal to each peak’s ideal distance (the average of its base point distances) in the direction towards the other peak. Then the mid-point between these two points is calculated. If this point qualifies as per the definition of a peak bridge point previously given, it is then placed on the list.

The ANNS local training ascent

The aim of the local training ascent is, starting from a base point, to find a series of **trail points** with increasing active set scores, and then select a point with the highest pending set score to represent the peak. The underlying principle is that there is a trend such that pending set scores will tend to increase along with active set scores up to a point where overtraining will reverse this trend. Since the training does not directly use the pending set, this statistically means that the peak net configurations are more likely to be of general use than if directly trained. A secondary aim is to provide a level of variability so that combinations of values for components can evolve, and also untoward effects of difficult or bad data records can, on occasion, be bypassed.

A simple implementation of the above aim would be very inefficient, which matters a lot because this procedure dominates the overall processing. Here a more efficient method is described which sacrifices a little accuracy for speed. Since the process is anyway repetitious, the loss of accuracy for individual ascents is more than compensated for by executing many more ascents. The loss of accuracy is also a gain in variability and is thus positively desirable!

The method must minimise the use of the most expensive activities. These are back propagation of active set records, and the scoring of the potentially much larger pending set. First we must deal with the step of training the current active set which is repeated to form the series of trail points on the ascent until the active set score fails to increase.

For each sweep, the current active set records are accessed in random order which is determined before any other processing is done. In general, within a sweep, each record is processed with reference to the previously processed record. The exception is the first record which uses the last randomly picked record for reference. The reference record is first back propagated. Where the record has just previously been back propagated and updated, a more efficient partial back propagation can be done to produce the exact same result as a full back propagation. The subject record must also be back propagated – it will become the next reference record after it is updated.

Overall in the sweep, the aim is to produce an update of one training step split over one or more components. Thus the target for a single record update is a substep of one training step divided by the number of records in the active set. The components encouraged for update are those that “benefit” the subject and and do not “disbenefit” the reference record. Thus statistically this tends to benefit the whole active set whilst avoiding the expense of testing every active record at every substep and introduces a controlled level of variability.

The detail procedure for each sweep is as follows :

The starting values for each component are remembered so that after the first update subsqequent updates of a component are restricted to those in the same direction. For the sweep, this ensures that the end point is not the starting point and encourages results towards the full training step away (spread across suitable components). The **default substep size** for each component is calculated by dividing the training step size by the number of records in the current active set.

The order of processing is decided randomly. The last record is designated as the reference and the first as the subject record. Both records are back propagated. Subsequently, the next record in the order chosen becomes the subject record and is back propagated. The previous subject record is redesignated as the reference record and its back propagation is updated to include superior components to the one component updated (if this occurred).

The process per reference/subject record pair is as follows :

If the score for each ouput node is perfect, then no futher processing occurs on the current pair. Otherwise, each component is examined in order to determine its suitability as a candidate for update. The process per component is as follows :

If, for the subject record, the component is inactive or the input quantity is within the ideal target range or equal to the (single) ideal target value, then the component is skipped. Also if the reference record is active and its input quantity is equal to the single ideal target value, the component is again skipped on the grounds that any update will worsen the score making the pairing incompatible.

Otherwise, for the subject record, the direction of update to achieve the target range or target value is determined and if the direction is opposite to that of any previous update for the component in question for this sweep, then again the component is rejected.

Yet another cause for rejection of the component is if the subject record direction of update is opposite to the reference record direction of update. Of course this does not apply if the component has not yet been updated during the sweep.

This leaves only those combinations where the component qualifies as a candidate. To determine the update quantity for the component and the weighting given to the candidate to qualify random selection between candidates the following steps are performed :

The **subject minimum update quantity** is calculated by taking the smaller magnitude of the component's default subset size and that quantity needed to modify the component to produce the nearest bound of its ideal target range, or the single target value as appropriate. The **subject maximum update quantity** is calculated by taking the smaller magnitude of the component's default subset size and that quantity needed to modify the component to produce the furthest away bound of its ideal target range, or the single target value as appropriate.

If the reference record is not active, then the **component update quantity** is set to the subject minimum update quantity and the **component selection weighting** is set to one. Otherwise, in a similar fashion as for the subject record, the reference record component target is used to calculate the **reference minimum update quantity** and the **reference maximum update quantity**.

If the subject minimum update quantity magnitude is equal to or less than the reference maximum update quantity magnitude, then the subject minimum update quantity is used as the component update quantity. Otherwise the component update quantity is set to the quantity with the smaller magnitude between the subject minimum update quantity and the reference maximum update quantity. If the reference maximum update quantity is lesser in magnitude than the magnitude of the subject minimum update quantity, then the selection weighting used is the ratio of the smallest divided by the largest of the two quantities. Likewise, if the magnitude of the reference minimum update quantity is greater than the magnitude of the subject maximum update quantity, the ratio of the smallest divided by the largest is used for the selection weighting. Otherwise the weighting is set to one.

When all the components have been processed, if one or more qualify for selection, one is selected at random using the weighting to modify the chance of being picked. The appropriate component update quantity is then applied and then the next pairing is processed.

If no component qualifies and no previous update has occurred for the specific sweep, then a number of attributes are accumulated across the sweep for each component. If, for each component, the subject minimum update quantity is not zero, either two positive attributes are updated or else two negative attributes are updated, each pair of attributes consisting of an **update candidate count** and an **update candidate accumulated quantity**. If, at the end of the sweep, no update has occurred, then one component update candidate is chosen at random from the combined positive and negative sets but weighted by the size of each update candidate count. Having chosen a component's update candidate count, its component quantity is updated in the associated direction by an amount, the lesser of the default substep size and the selected update candidate accumulated quantity divided by the product of the corrersponding update candidate count and the number of records in the active set. Thus having guaranteed one update, the sweep is then repeated to give the probability of further "compensating" updates.

At the end of a sweep the active set score is calculated. If the active set score is perfect (unlikely) then the sequence of sweeps is terminated. If the score has improved then the trail point configuration is recorded and a new sweep is undertaken.

[Note that incoherent ranges will statistically reduce updates of the associated components. But this does not eliminate unravelling of such incoherence via updates of other components and statistically leads to solution via earlier structural expansion.]

If the first sweep is unsuccessful then the **confirmed peak** point is assigned coincident to the base point, the pending set score for that point is determined and no further processing is undertaken for this ascent. Otherwise an attempt is made to recover from a "stumble" and resume the ascent.

If the last trail point has an inferior active set score to the previous, and a previous sweep was successful, i.e. had an improved active set score, then a projected configuration is calculated and scored for the active set. If that projection was an improvement upon the previous successful trail point, then the last trail point is jettisoned and the ascent process continued from the projected trail point, thus enabling the "stumble" to be efficiently corrected. The projection is achieved via consideration of the last three trail points. The size of the projection, in training units, is calculated from the sum of the difference between the components of the latter two configurations. The projection extends the line from the first of the trail points to and beyond the second. The direction is given by the difference in component values between the first two trail points. If the resulting projected point is coincident with the third point, then the trail is deemed complete. Otherwise, the active set is scored for the projected configuration and if this is an improvement on the last successful step active set score, then the sweeps are resumed.

[TODO : add a refinement sweep to search for small overall improvement in one component]

Having completed the trail, the next task is to efficiently find the trail point with the best pending set score. The idea is to minimise the points examined in this way, without too often missing the best point. This process starts with the selection of three points from the continuous line determined by using intermediate coordinates between trail points. This line forming a (non-Cartesian) horizontal dimension is plotted against active set rating for a vertical dimension to give a two dimensional graph.

For the very first time the centre of the trail is assigned to the selected centre point. The active set rating for this centre point is estimated as proportionally intermediate according to the distances (in training units) between preceding and succeeding trail points. This rating initialises an **history centre point variable** used for the next training ascent when calculating the next selected centre point. Again, for the very first time, the trail base point is assigned as the left bound selected point and the trail end point is assigned to the right bound selected point. The distance (in training steps) between the base and end points of the trail initialises an **history range variable** used for the next training ascent when calculating the next left and right selected bounds. Both history variables are modified once the peak point is found (see later).

The three selected points on subsequent training ascents are calculated normally by using the history variables mentioned in the previous paragraph. However, in the case of a short trail range of two or less training steps, the centre of the trail is assigned to the selected centre point and the extremes of the trail are assigned to the selected bounds. Otherwise, if the history centre point (as calculated by estimated active set rating) lies on the trail line at least one training step from either end, this point is assigned as the selected centre point. If however the history centre point is closer than one training step to a trail extremity, then it is adjusted to exactly one step away from that extremity. The left and right selected bounds are then allocated by referring to half the history range. If a bound goes beyond a trail extremity it is adjusted to that extremity.

Having selected the three points, the pending set is scored for each and the pending set ratings are calculated. The selected trio gives rise to one of four results :

1. All three points have exactly the same rating (unlikely).
2. The centre point has a higher rating than both bound points.
3. The centre point has a higher rating than one bound and exactly the same as the other.
4. One of the bound points has a higher rating than the other two points.
5. Both bound points have exactly the same higher rating than the centre point.

The 5th case is treated as for the 4th case (see next) but treating the left bound is if it were higher than the right.

In the 4th case, if the high bound point is not at an extremity, then an attempt is made to extend the bound by the distance between it and the current selected centre point. Of course, if it would exceed the trail extremity it must be truncated. The new point is assigned as the appropriate bound, the high bound point is assigned as the selected centre, and the previous selected centre as the other bound. The new point is scored for the pending set and the process then loops back. However if the high bound point is at an extremity, a new selected centre point is assigned halfway between the high bound and previous selected centre point relegating the previous selected centre point to become the other bound. Again, the new point is scored for the pending set and the process loops back unless limitation criteria are met (see later).

If the centre is the higher of the three selected points (2nd case), then an attempt at curve fitting is made to predict where the peak might be. By assuming a smooth rate of increase and decrease of the pending set score it is possible to radically improve on results given by intermediate straight line projection. The formula used has the following terms:

x_L, y_L are the training step coordinates for the left bound, L.

x_R, y_R are the training step coordinates for the right bound, R.

x_C, y_C are the training step coordinates for the selected centre point, C.

n is an arbitrarily large number, say 64,000, which divides CL and CR to aid calculation.

I_1 thru I_n are the increments between C and L to form the estimated curve on the left side.

J_1 thru J_n are the increments between C and R to form the estimated curve on the right side.

The left initial increment, I_1, is set to extend the right hand gradient, thus equal to $(y_R - y_C) / n(x_R - x_C)$.

The right initial increment, J_1, is set to extend the left hand gradient, thus equal to $(y_C - y_L) / n(x_C - x_L)$.

u is the left side modifier applied at every increment such that $I_n = I_1 + nu$.

v is the right side modifier applied at every increment such that $J_n = J_1 + nv$.

s is the number of iterations between C and the left side peak P and is equal to I_1/u.

t is the number of iterations between C and the right side peak Q and is equal to J_1/v.

x_P, y_P are the coordinates for left side peak, P when $I_s = 0$.

x_Q, y_Q are the coordinates for right side peak, Q when $I_t = 0$.

Using the formula for the summation of an arithmetic series, u and v can be determined thus:

$$y_L - y_C = (I_1 + I_n)n / 2 = (2I_1 + nu)n / 2$$

Therefore $u = ((2(y_L - y_C) / n) - 2I_1)n$

Similarly $v = ((2(y_R - y_C) / n) - 2J_1)n$

Again using the formula for the summation of an arithmetic series, y_P and y_Q can be determined thus:

$$y_P - y_C = (I_1 + I_s)s / 2 = I_1^2 / 2u$$

Therefore $y_P = y_C + (I_1^2 / 2u)$

Similarly $y_Q = y_C + (J_1^2 / 2v)$

If y_P is equal or larger than y_Q then P is selected, else Q.

The x coordinate of the selected point is calculated thus :

$x_P = x_C - (x_C - x_L)s / n$ or $x_Q = x_C + (x_R - x_C)t / n$

If the centre point is higher than one bound but exactly equal to the other (3rd case), the above method is confined to one side only. If the centre point is higher than the left bound then the right side x_Q is calculated. If, however, the centre point is higher than the right bound then the left side x_P is calculated.

When all three points have exactly the same active set ratings, a point midway between the left bound and centre point is chosen.

The selected point is then scored against the pending set and unless limitation criteria are met it forms part of the next a selected trio for the next iteration of locating the candidate local peak. If the new selected point is not lower than both adjacent processed points, then it forms the centre of the trio. Otherwise, if the left adjacent point is higher and is not at an extremity, this becomes the centre and thus the new selected point becomes the right bound. Similarly, if the right adjacent point is higher and is not at an extremity, this becomes the centre and thus the new selected point becomes the left bound. Where extremities curtail an adjacent point becoming a centre, the new selected point becomes the centre even although it is lower than the bound.

There are two criteria which both must be met in order to end the loop processing selected trios as detailed above. First, the distance between the highest point of the selected trio must be less or equal to half a training step. This seemingly arbitrary figure is chosen to ensure (just) adequate examination of the curve between all relevant trail points. The second criterion is that there is no immediately previous candidate local peak such that the increase in rating for the new local peak (if there is an improvement) is greater than half the previous. This allows significant improvement with sub-training step sizes but is guaranteed to be self-limiting.

The above method can be modified in order to save some of the active set back propagation processing. The active set sweep loop can be “paused” once the active set rating reaches or just exceeds the upper bound as calculated by the history centre point variable plus half the history range. When processing the pending set, if a higher upper bound is

needed, the active set sweeps must be resumed to fill in the required range of values. (This is explained separately as a modification in order to simplify the description of the main method).

Once the local peak is found, the history variables are updated. The history centre point variable is reset to the midpoint between the previous value (the starting centre point) and the candidate local peak position (the finishing point). The history range variable is set to the average of the previous value and twice the difference between centre starting point and candidate local peak position.

The next task is to potentially adjust the training step size. This need only be considered once every four training ascents. If during the previous four ascents, one or more ascent trails were less than 3 training steps from base point to local peak, then if a minimum limit is not exceeded, the training step size for each component is reduced by one quarter. The reduction is suppressed if this would reduce the step size under one quarter of the average training step size accumulated from the start of training expeditions for this net configuration. The minimum limitation allows progressive alteration of initial training step sizes in line with an overall trend.

When considering upward adjustment of training step size, the criterion is that the trail length all four of the previous four ascents must be over six training steps. If this criterion is met the training step size of each component is increased by one quarter. After upward adjustment of training step size, the existing peaks may dip below three training units from their nearest neighbour. However this does not create additional processing. In particular, the list of peak bridge points is left untouched.

The next task, is to potentially merge the candidate local peak with nearby peaks. An iterative process begins with identifying the closest existing peak. If none is closer than three training steps, then the merge process terminates. If two or more peaks are exactly equidistant then one peak is chosen at random. A centre point on a line between the two peaks is then found and scored against the pending set. The trio of points is then processed exactly as in the process searching for the local peak in order to find the highest point along that line. (The new *merged* peak can of course be coincident with either one of the two subject peaks.) This merged peak then replaces the two subject peaks and is allocated to the current base point and also to the base points of the now defunct selected nearby peak.

When a merged peak is formed, the above process is also applied to it. The process repeats until no peak exists closer than three training steps from any other.

If the new peak is higher than previous peaks, the best net update process is invoked.

The final task to perform is to merge the trail should the new candidate peak be close to the base point of an existing trail. This is quite unlikely but can theoretically happen in featureless or complex solution attribute topologies - in which case merging trails, and thus increasing trail length, helps in expanding the distance at which new base points will be created to further the search. There are two criteria that must both be met before joining the candidate peak to an existing trail base point. First, the candidate peak must not be higher than the existing base point. Second, the candidate peak must not be more than one training step away from the existing base point. If more than one base point qualifies, then the nearest is used. If equidistant, a qualifying base point is chosen at random. the trail is merged by replacing the existing trail’s base point with the current trail’s base point.

Verifying and initialising the data

Once the schema has been defined and all the input data presented, a process is undertaken to report on logical inconsistencies and, by default, to automatically remedy these.

For line and square grid array nets, a **normalisation** process is first undertaken to present array data in an order to make it easier for duplicate records (considering symmetry) to be found. To do this individual values must be determined for heap logical offset arrays elements and, for other array types, each record's array orientation indicator. If more than one array secondary node is present an ***arbitrary*** **array precedence order** is established. The actual order of arrays does not matter but when normalising one specified order of some sort is required.

For heap arrays the heap logical offsets are initialised to physical within node content precedence order.

For other types of array, for each record, the elements across multiple arrays (of same type and size) are accessed in arbitrary array precedence order within logical offset order for each candidate array orientation indicator value (1 thru 8 for square grid, 1 and 2 for line array). Any combined content greater than the lowest combined content eliminates the corresponding candidate array indicator value. The processing finishes when only one array orientation value remains, or if all the logical elements are processed, in which case the lowest remaining value is selected for the array orientation indicator.

The first type of error is when all the ouput nodes for any one input record are "don't care", i.e. posses an infinite target range. The user is asked to supply the missing information. By default, the record or records concerned are deleted. For arrays, all elements must be "don't care" for the record to be erased. Partially filled array information is acceptable and the user is not asked to fill in "don't care" elements.

The second type of error is when two or more records with the exact same input node values give differring output target ranges. (Note that null input data is deemed to always match non null data.) To check this the entire internal set of records is sorted on input node content with an arbitrary precedence order of input nodes. The user is asked to either change the data, or add input nodes with different input data values to discriminate between the varying cases. The default is that the output ranges are set to ranges encompassing the different targets, thus producing identical duplicate records. All duplicate records are automatically merged whilst accumulating their record replication factor conent. In the case where all outputs are set to infinite target ranges, the records are deleted.

The third type of error is when an input value falls outside of the stated input range. The user is invited to change either the input value and/or the input range. By default the input range is changed to fit.

The next task is to use the entire internal set to determine **population factors** for each individual piece of target data relating to each singleton or element group of an output node. Population factors are used in scoring working nets (see "Scoring ANNS working nets" section). They (selectively) reduce the training effect of repeated content which otherwise might dominate the training process. The formula for each factor is :

$$P = A / U$$

where

- A is the average number of **population counts** for each element or singleton output node target.
- U is the individual population count for a specific element or singleton output node target

The population count for each element or singleton is achieved by counting the number of record intances in which the target value range overlaps the subject range, across all the element group (or just the singleton), across the whole internal set. Note that this does not materially invalidate the independence of the reference set, because only popular population factors are used thus giving an approximate parity between load and reference sets. The reason for using the whole internal set is that this allows score comparison between nets with differing load sets.

The ANNS initial bedding phase

It is often the case that many of the input nodes presented in a neural net turn out to be superfluous. The volume of processing associated with this superfluous input can not only delay the production of the desired quality of result but also, by dint of its sheer weight make it in unattainable even within generous time constraints. To counter this, the main task of the initial bedding pass is to select a very small section of the input that is statistically likely to produce significant results with a compact net structure. The selection process is intended to be relatively efficient albeit approximate as compared with the main training process which has the major constraint of achieving accurate results. The trick is to find simple characteristics in the input data that indicate an increased likelyhood that *simple* connections between selected input nodes and the output nodes will produce better than random results. If no such characteristics can be found, then a random selection still has the benefit of being likely to reduce the volume of overall training computation. This is owing to the fact that it is more efficient to process a small net structure and gradually expand it, than start with a larger structure and gradually reduce it.

For the initial net, the desired *minimum* simple net structure is defined as one that, for *each* output node element, is linked to one **primary** input node *or* one **derived** input node. Derived input is created by (automatically) connecting the training data to existing ANNs and using the output to form new input to the new net structure. (How the matching works is described in the “Derived input” section.) Additional links to the input constant node and further input and derived input nodes may be added if it is estimated that this is likely to contribute significantly to the potential quality of the first phase training outcome, outweighing the efficiency "cost" of processing the extra links.

There may be very large amounts of derived input. This gives rise to the slight possibility of selecting the input which *by mere coincidence* matches the targets. This possibility is dramatically reduced by using only a subset of the load set (chosen at random) for selection purposes. If a perfect or near perfect match should be found by coincidence for this random subset then the remaining part of the load set is still available to act as a training corrective. If this remaining part is also a perfect or near perfect match, this is far less likely to be through coincidence because it was not specifically selected for. Because of the potential for a large number of derived input permutations, using a subset for selection has the added bonus of proportionally speeding up the selection processing.

It is not possible to precisely determine the suitability of input and derived nodes for training. This can only be revealed in hindsight by the success or otherwise of the training. However, there a few features which indicate a statistical probability of reduced suitabilty. Duplicate input content, that is not matched by duplicate output targets, forms a logical contradiction, and means that further, potentially inefficient, net structure to connect to other input nodes is needed in order to resolve the conflict. In the case of derived input nodes, because every possible combination is automatically tried, such **contrary duplication** is very likely to be caused by an inappropriate derivation.

Where both input and output are of numerical type, a good indicator of some simple significant relationship is when the order or reverse order of numerical values match more closely than a random distribution would.

A similar comparison can be made when either the output or input is numeric but the corresponding input or output is of binary or indicator/identifier type. The records are sorted in numerical order, input or output as appropriate. An attempt is made to rank each binary element so that the primary node can then be interpreted as numeric, thus reducing the comparison to the same as when both input and output nodes are numeric.

Where both input and output are binary or indicator/identifier types, a simple structural relationship is indicated by one input element or its negation matching an output element target to a greater extent than might be predicted for random input. A refinement is to see if other elements, which are less directly matched to the output, can be used to improve on the original match, again comparing the results against random input.

The loose basis for comparing the different suitabilities, given differing input types, is to use a random distribution of input values as a baseline of suitability. The calculation of this baseline can be done by sampling, which has the added advantage of giving a desirable degree of variability. (Variability reflects the limitations of accuracy of the method used and gives the possibility of finding a different better solution by merely repeating the process.) This will become clearer in the detailed method described below.

Direct and derived binary input are treated both as binary and numeric class, i.e. as if declared as unsigned numeric, and are selected as one or the other for the purposes of appropriate connections according to the value of the calculated **suitability indicator**. Direct and derived signed number input is examined both for an absolute connection and for a magnitude connection. Again the suitability indicator is used to select the appropriate type of connection made.

In principle, every *primary* input node and every validly derived input node is examined against each *primary* output node for a selected set of records. Typically half the load set is selected at random. But if the load set only has fifteen or less records, three are reserved and the rest form the **selection set**.

The same selection set can be used for multiple output nodes. Otherwise the following processing is separate and independent for each primary output node and, in the case of multiple binary secondary nodes, for each binary element.We shall consider selection of suitable singleton input nodes first, and then extend the description of the required processing to cover arrays.

The inputs are examined in a random order and each is processsed to obtain a **suitability indicator**. This indicator for each output element, notionally and aproximately expresses the potential fraction of a perfect solution above a neutral baseline, divided by the number of links involved (ignoring constant node links), thus enabling the suitability of each input for inclusion in the initial constructed net. The detail processing to calculate this indicator depends on the combination of input and output primary types. Primary binary, indicator and identifier types are all implemented via secondary binary nodes and are referred to as **binary class** to distinguish them from primary numerical node types, i.e. signed and unsigned numeric.

Binary class output nodes are examined against both binary class and numerical input nodes. Against binary class input each output element is individually examined. But against numerical input an attempt is made to select output elements and order them so as to form a binary number which can then be treated as if it were a single numerical output node. This translation requires detail explanation :

The selection set is first sorted into random within ascending numerical input order. Then each output element is examined in order to potentially select the most significant one. The selection set records are sorted into ascending order according to the numerical input under examination. For each binary element, this produces an output binary string (using the target values), made from one bit for each record. The simplest most significant result would either be all ones first, or all zeroes first. The norm is a more dispersed set of ones and zeroes. So the task in hand is to evaluate whether and how much the distribution is more ordered than random. The first step is to sum the positions of the ones to find the average ones position and separately sum the positions of the zeroes to find the average zeroes position. The lowest average, L, is then noted. For example the string 0110001101 gives (2+3+7+8+10)/5 and (1+4+5+6+9)/5, i.e. 6 and 5 of which the latter is chosen. Because, in this case, the latter average was chosen, if the input node ends up being selected, the rev-binary secondary output node would be used for the initial structure rather than the binary. (If the averages are equal, selection between rev-binary and binary seconary output nodes is random.) The next step is to find a random distribution median giving a baseline value, B. To this end the process is repeated as many times as there are selection set records (S), each time selecting the order of the string bits randomly and producing a lower average position P_n. For even numbers of selection set records, the average of the the mid-value records is taken. If L is less than B, the element is a candidate for selection. From the candidates, the element with the lowest L is chosen as the most significant, ties being resolved by random choice.

The next significant element is chosen from the remaining output elements using a similar process. However, the process must take into account the already chosen elements. This is done by formings a composite binary number and creating output strings split into substrings within each value. For example, assuming that an element is selected as in the example from above, a new string 1011000100 will be split into 1..100..0. matching the zeroes from the first element string and .01…01.0 matching the ones. The zeroes and ones averages are now taken from the positions in both the substrings. The ones average is thus (1+2+2+4)/4 or 2.25 and the zeroes average is (3+4+5+1+3+5)/6 or 3.5 resulting in L equal to 2.25. The random baseline is also calculated using the substrings but selecting zeroes and ones from the new output element values in random orders. Subsequent element selection will result in more fragmented substrings. In the example a further element will result into upto four substrings matching the positions selected by the composite number values 0, 1, 2 and 3 or in binary 00 01 10 and 11 where the first element string contributes the most significant bit and the second element string the least. In the example the two selected elements give a composite string 1 2 3 1 0 0 2 3 0 2. A new output element string 0110011101 will thus be split into four substrings - ….01..0. and 0..0…… and .1….1..1 and ..1….1.. .When a resulting substring contains all zeroes or all ones it is ignored . In this case only one substring is of interest giving averages of (1+3)/2 giving L equal to 2. If no substring qualifies then the element is eliminated. The process is complete when no further candidates are found. If the process results in the selection of more than one element, the suitabilty indicator is calculated using the composite numbers treated as target data for a notional numeric output node. Singleton binary output nodes are treated merely as a subset of the multiple element case with the selection process described above being used to determine between use of the binary or rev-binary secondary output node. Whether single or multiple, the selected rev-binary and/or binary elements are then treated as a single numerical output when evaluating the suitability indicator.

When the input is binary class and the output is numerical a similar process is applied to select multiple input elements. Instead of ascending input order, the selection records are sorted into ascending output order (using the mid point for target output ranges). The same process is then applied to the input substrings to form a set of composite numerical values. The lowest average for one and zero positions is noted. If the lowest average is from the zero position total, if selected the input element uses the secondary rev-binary node when the structure is initialised. If the process results in the selection of more than one element, the suitabilty indicator is again calculated using the composite numbers treated as target data for a notional numeric output node.

Except for when both input and output are of binary class, we can now treat input and output both as numerical values. For binary and unsigned numeric output, the magnitudes of both input and output nodes are used in the selection process. For signed numerical output, the same process using the magnitudes is used but an additional process using absolute values is performed and overrides the former process if the suitabilty indicator proves equal or greater.

The “unsigned” process, the **magnitude process**, starts with the allocation of **output position ranges** to each selection set record. Sorting the records into ascending lower bound ideal output target value order improves the efficiency of this task but is not logically essential. The lower bound for the output position of any one record is defined as one plus the number of other records containing lesser *upper* bound values. The definition of the upper bound of the output position for any one record is the number of other records containing a lesser or equal *lower* bound value.

The second task is to achieve **ascending and descending input positions** for each selection set record. The records are first sorted into ascending input value. Each record is then given an **ascending input position**. Where records share the same input value, they are deemed to share the same position which is given the numerical value of the average of the shared places. For example records occupying fourth and fith places with the same input value would be allocated 4.5 as the average. The third task is to repeat this process but with the records sorted into descending input value order in order to calculate the **descending input positions**.

The third task is to find the least value between the ascending and descending summed **discord difference**. For each record the discord difference is defined as an absolute measure of difference between the input position and the bounds of the output range. If the input position lies within the output range, the discord difference is zero. If it is lower than the lower bound, subtracting it form the lower bound gives the desired value. If it is higher than the upper bound, subtracting the upper bound from the input position value gives the desired number. Summing this numbers for ascending input positions and then descending input positions gives two totals. The lesser of the two totals, forming the **actual discord total**, is chosen. If the descending discord total is less than the ascending, then if the input is eventually selected, the structure will use the reverse secondary output nodes rather the standard in order to simplify linkage. If the totals are identical the choice of reverse to standard is randomly made.

The fourth task is to calculate the **baseline discord total**. The calculation of the discord difference and selection of the least between ascending and descending sums is repeated as many times as there are selection set records, but with random matching of the input values to selection set records. The median selected discord total then forms the baseline. The highest selected discord total forms the **upper bound discord value**. The lowest selected discord total forms the **lower bound discord value**.

The suitability indicator is calculated by dividing the difference between the actual and baseline discord totals by the difference between lower and upper bound discord values. If the baseline discord total is lower than the actual discord total the suitability indicator is set to zero. Otherwise if the number of unique input values is less than the number of non-overlapping output position ranges, then the suitability factor is further reduced by multiplying it by this ratio. For input formed of multiple binary elements, each element is assigned a fraction of the suitabilty indicator value according to the number of elements selected. For output formed of multiple binary elements, for each element the suitabilty indicator value is halved to reflect the more complex structure needed (see later).

The above process for calculating numerical suitability is illustrated by the following worked example:

Selected Discord Table

Reference Position	Ideal Output Target Value	Output Position Range	Input Value	Ascending Input Position	Descending Input Position	Ascending Discord Difference	Descending Discord Difference
1	1-3	1-4	6	3	8	0	4
2	2-4	1-4	18	5.5	5.5	1.5	1.5
3	3-3	1-4	10	4	7	0	3
4	3-9	1-9	25	8	3	0	0
5	5-7	4-8	30	9.5	1.5	1.5	2.5
6	6-8	4-8	2	1	10	3	2
7	7-7	4-8	18	5.5	5.5	0	0
8	7-10	4-10	3	2	9	2	0
9	9-10	4-10	24	7	4	0	0
10	10-10	8-10	30	9.5	1.5	0	6.5
						8	19.5

Baseline Discord Table 1 of 5

Output Posn Range	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference
1-4	1	10	0	6	3	8	0	4
1-4	3	8	0	4	2	9	0	5
1-4	9.5	1.5	5.5	1.5	7	4	3	0
1-9	2	9	0	0	1	10	0	1
4-8	7	4	0	0	8	3	0	1
4-8	9.5	1.5	1.5	2.5	5.5	5.5	0	0
4-8	5.5	5.5	0	0	4	7	0	0
4-10	5.5	5.5	0	0	9.5	1.5	0	2.5
4-10	8	3	0	1	9.5	1.5	0	2.5
8-10	4	7	4	4	5.5	5.5	2.5	2.5
			11	19			**5.5**	18.5

Baseline Discord Table 2 of 5

Output Posn Range	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference
1-4	5.5	5.5	1.5	1.5	2	9	0	5
1-4	5.5	5.5	1.5	1.5	9.5	1.5	5.5	0
1-4	9.5	1.5	5.5	0	5.5	5.5	1.5	1.5
1-9	9.5	1.5	0.5	0	3	8	0	0
4-8	4	7	0	0	9.5	1.5	1.5	2.5
4-8	2	9	2	1	8	3	0	1
4-8	8	3	0	1	4	7	0	0
4-10	3	8	1	0	7	4	0	0
4-10	7	4	0	0	1	10	3	0
8-10	1	10	7	0	5.5	5.5	2.5	2.5
			19	**5**			14	**12.5**

Baseline Discord Table 3 of 5

Output Posn Range	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference
1-4	3	8	0	4	7	4	3	0
1-4	8	3	4	0	1	10	0	6
1-4	1	10	0	6	8	3	4	0
1-9	2	9	0	0	2	9	0	0
4-8	9.5	1.5	1.5	2.5	9.5	1.5	1.5	2.5
4-8	7	4	0	0	9.5	1.5	1.5	2.5
4-8	4	7	0	0	5.5	5.5	0	0

4-10	5.5	5.5	0	0	5.5	5.5	0	0
4-10	5.5	5.5	0	0	4	7	0	0
8-10	0.5	1.5	0	6.5	3	8	5	0
			5.5	19			15	**11**

Baseline Discord Table 4 of 5

Output Posn Range	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference
1-4	5.5	5.5	1.5	1.5	4	7	0	3
1-4	2	9	0	5	1	10	0	6
1-4	8	3	4	0	7	4	3	0
1-9	7	4	0	0	2	9	0	0
4-8	3	8	1	0	3	8	1	0
4-8	5.5	5.5	0	0	5.5	5.5	0	0
4-8	4	7	0	0	9.5	1.5	1.5	2.5
4-10	9.5	1.5	0	2.5	9.5	1.5	1.5	2.5
4-10	1	10	3	0	8	3	0	1
8-10	9.5	1.5	0	6.5	5.5	5.5	2.5	2.5
			9.5	15.5			**9.5**	17.5

Baseline Discord Table 5 of 5

Output Posn Range	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference	Random Asc Input Posn	Random Desc Input Posn	Asc Discord Difference	Desc Discord Difference
1-4	4	7	0	3	9.5	1.5	5.5	0
1-4	1	10	0	6	2	9	0	5
1-4	7	4	3	0	5.5	5.5	1.5	1.5
1-9	5.5	5.5	0	0	1	10	0	1
4-8	8	3	0	1	4	7	0	0
4-8	2	9	2	1	8	3	0	1
4-8	5.5	5.5	0	0	7	4	0	0
4-10	9.5	1.5	0	2.5	3	8	1	0
4-10	3	8	1	0	9.5	1.5	0	2.5
8-10	9.5	1.5	0	6.5	5.5	5.5	2.5	2.5
			6	20			**10.5**	13.5

This gives 10 selected values - 5, 5.5, 5.5, 6, 9.5, 9.5, 10.5, 11, 11 and 12.5. Since there are an even number the median is taken as the average of the ranked 5th and 6th positions, ie. (9.5 + 9.5)/2 or 9.5. This baseline discord total is higher than the actual discord total indicating some suitability. The suitability indicator value is (9.5 – 8) / (12.5 – 5) or 0.2 (dividing the difference between the baseline value and the actual value by the difference between upper and lower bound values).

The “signed” version of numerical selection, the **absolute process**, is similar to the “unsigned” or magnitude process, but with the following differences:

1. It applies only for signed numerical output.
2. The abs values for both input and output are used when evaluating actual and baseline discords.
3. A target range extending from below zero to above zero is translated into an absolute range starting from zero to the greater of the absolute values for the lower and upper bound.

When both input and output are of binary class, each secondary binary output node is treated independently. For each secondary output binary node each secondary binary element within each primary input node is examined for a data

match for each record in the selection set. A **match string** is created with ones where the data is identical i.e. both zeroes or both ones, and with zeroes where there is a mismatch. If there are more zeroes than ones, the match string is negated. (If finally selected, and if the match string was negated, then the output element chosen for the structure must be rev-binary.) The number of ones in the resulting match string is noted. For example the input string 10111010 compared with the output string 01110001 gives 00110100 which, because there are more zeroes than ones, is negated to give the match string 11001011 thus counting five ones, the **match string concordance count**. A **baseline concordance count** and **lower** and **upper concordance counts** are achieved by taking the median and extremeties of a number of random samples (say the selection set size), each sample made by allocating the input data order randomly and applying the same processing. If the baseline concordance count is lower than the match string concordance count this indicates some suitability. A suitability indicator for the individual binary input node is then calculated by dividing the difference between median and actual values by the difference between upper and lower bound values. This process is repeated for each secondary binary node owned by the primary node under examination and the most suitable is chosen, or if more than one element share the same suitability indicator value, this is resolved by chosing one at random. If the suitability indicator has a value of exactly half or over, no further selection is required because the overall efficiency would probably be reduced by adding more links. But, if the suitability indicator value is less than half, the process is repeated but using the current match string to compare with the input strings. If the best input string now has a suitabilty factor equal or greater than the first selected input element, the new element is additionally selected. If additional selection occurs, the iterative process is repeated but comparing the new suitabilty indicator values with the *average* of the already selected suitabilty indicator values. When the new suitabilty indicator fails to reach half the required value, the process is terminated. The average suitabilty indicator value is taken forward for comparison with other input node suitabilty indicator values.

The method of calculating the suitability indicator for input connected to output array topologies is a development of the method used for singletons. This applies regardless of whether the input has a matching topology, i.e. singleton input to line and square grid output, line and square grid input to heap – all are encompassed by the method dictated by the output type. The additional processing is described below.

The heap because it has no connection between elements, is the simplest to deal with. Each element is treated as a separate record, thus creating a notional selection set the size of the sum of all the elements, or at least all the elements except those excluded by “don’t care” target ranges. The overall suitability factor is thus taken as the average of each suitability factor calculated for each element.

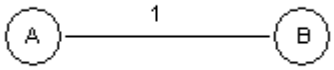
The selection set records for output line arrays are split into groups according to symmetry, i.e according to shared l-coordinate content. Each group forms a notional selected set. The aim is to find the best group in terms of suitability and select this group regardless of the suitability of the other groups. This is because evolving the net for one symmetry group is a progressive step towards evolving the net to cater for all symmmetry groups. The overall suitability factor for each group is taken as the average for all the elements in that group and then multiplied by the ratio of content present (as opposed to don’t care) elements to the total number of elements in the specific group. The key suitability factor is that of the highest scoring group. The adj[1] and adj[2] nodes are treated at this stage as two separate array inputs, again split into groups according to symmetry, but with halved suitability indicator values in order to reflect the added complexity of the structure that would be required to implement their selection.

Ouput square grid arrays are treated in a similar fashion to output line arrays. In this case elements are grouped according to each unique value for the combination of l-coordinate and d-coordinate content. Again, each group suitability factor is modified according to the proportion of content present elements. The adj[1] thru adj[8] nodes are treated in a similar fashion to the adj[1] and adj[2] nodes for lines, that is as eight separate inputs with halved suitability values grouped according to symmetry defined by combination of l-coordinate and d-coordinate content.

Once the input node has been selected, the appropriate structure is created and initialised depending on the selection and output combination. The initialisation crudely attempts a plausible set of component values in order to speed up the overall process, but the training process is relied on to find peak values and if the values were fortuitously to be set at a local peak, then the training process is expected to find other peaks. All binary exponent components are fixed at zero and all other exponents are initialised to one. The other initialisation details for each selection / output combination are as follows :

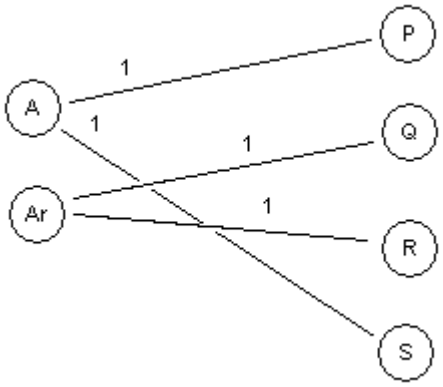
Single binary input element to single binary output element

[anns014.gif]



The above diagram shows the simple structure needed for a binary input to binary output node. The AB link is fixed at a weight of 1. Since the exponents are also fixed at zero, this structure is untrainable. However it can be used in combination with trainable d-coordinate and/or l-coordinates connections (see later) where the B node is a square grid or line array (see later). If used in isolation, the training session will be a short one but, assuming perfection is not attained, this will serve to initialise the next expansion phase where further trainable links are added. The B node can also represent a secondary sign output node where absolute connection is selected (see later). The A node is either the binary or rev-binary secondary node depending on the selection process.

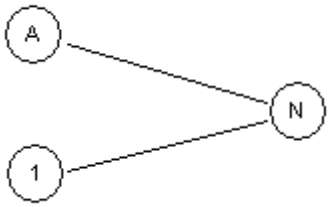
Single binary input element to multiple binary output element



[anns015.gif]

The above diagram shows the structure that might occur for selected output binary elements (which therefore do not take place in a pseudo-numeric selection). Taking each output element separately, the structure is identical to that of a single binary element to single binary element structure depicted in the previous diagram. The choice of connection to the binary or rev-binary secondary input node is determined at the selection process. Of course the individual binary output elements could be connected to other binary class input elements.

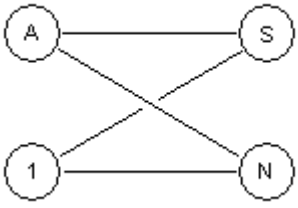
Single binary input element to unsigned number output



[anns013.gif]

The above diagram shows a binary or rev-binary secondary node, as appropriate to the selection, connected to the abs secondary number node. The AN weight is initialised to the median N (magnitude) value. The 1N link is initialised to zero.

Single binary input element to signed number output



[anns016.gif]

The above diagram shows a binary or rev-binary secondary node, as appropriate to the selection, connected to the sign and abs secondary number node. The AN weight is again initialised to the median N (magnitude) value as is 1S weight. The AS weight and the 1N weight are initialised to minus the median N value.

Multiple binary input element to single binary output element

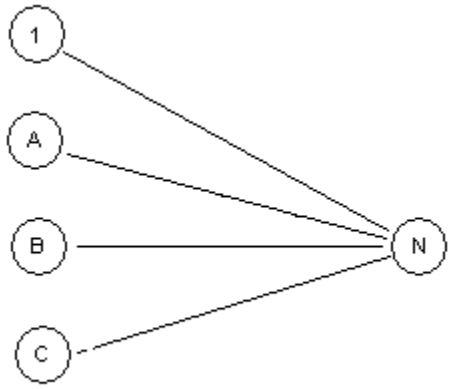
Only one input element is selected, so the structure used is as for a single input element – see before.

Multiple binary input element to multiple binary output element

Each binary input and output element is treated separately – see before.

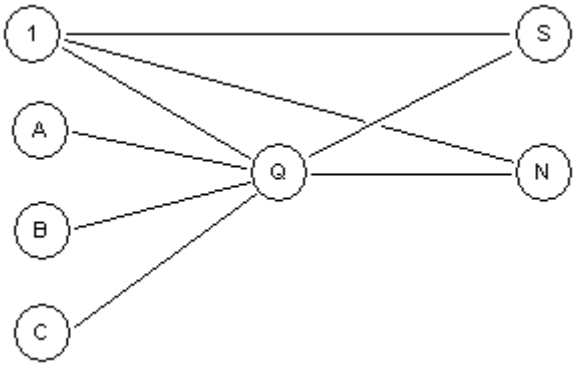
Multiple binary input element to unsigned number output

A single element may be selected according to the suitability indicator value in which case it is treated independently – see before. Otherwise the binary class input may be selected as a pseudo-number in which case the structure is created as follows:
[anns017.gif]



Either the binary or rev-binary secondary node is chosen according to the selection process. All links are initialised to zero leaving the training process to sort out the appropriate weightings.

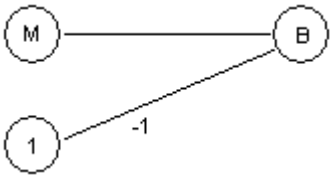
Multiple binary input element to signed number output
[anns009.gif]



The above structure is designed to produce an intermediate magnitude at node Q. 1S is initialised to a weight equal to the mean magnitude value for N and 1N is initialised to minus the same quantity. AQ, BQ and CQ are initialised to zero letting the training process sort out the relative importance of each element.

Signed/unsigned number input to single binary output element

[anns018.gif]

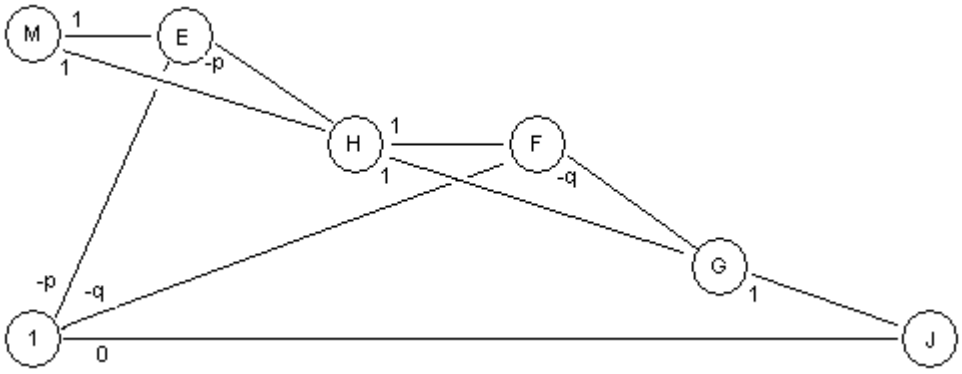


The mag or rev-mag secondary input node as dictated by the selection process, is connected to the binary output initialising the MB link to the inverse of the mean magnitude value of M. A link to the constant node is initialised with a –1 weight.

Signed/unsigned number input to multiple binary output elements

Each singleton binary output element may have both a binary input and a numerical input node selected. In this case the selection between the two is made in favour of the numerical input if its suitability factor is equal or better than the average suitability factor of all the selected input binary elements that match the output elements that share the numerical input. Assuming the numerical input is selected, the relevant output elements are no longer considered for connection to any selected binary input elements. Instead, a structure is created to give a cascade remainder effect to translate the numerical content into a binary form. The translation is done by training and encompasses possibilities other than a straight binary equivalence. The structure can be understood from the following example :

[anns008.gif]

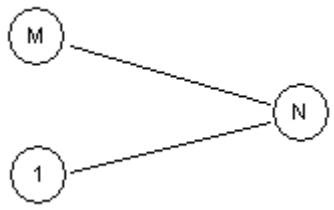


In the above diagram, depending on selection (see before) the mag or rev-mag node M provides the variable input content. The binary output nodes, E, F and J all have zero exponents to produce only binary output. The intermediate nodes H and G hold remainder values carried forward to the next cascade level. The links initialised with a link weight of one are all fixed. The value p is the median value of input content. The value q is the median value of input after p is subtracted where the content is equal or greater than p. These negative of these values are used to initialise the links shown and will be subject to training. The J node contains any residue, but this may be modified by the zero initialised 1J link during training.

Note that links with similar values at the end of the training process will be rationalised so that they concur, thus for example the divergence of the 1E and EH weights will be automatically corrected if appropriate.

Signed/unsigned number input to unsigned number output

[anns011.gif]

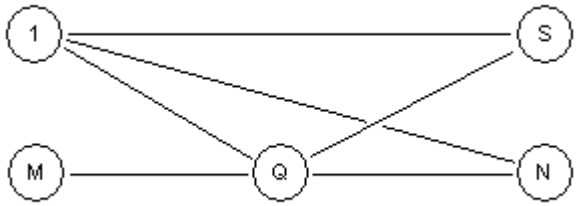


The above diagram shows the mag or rev-mag secondary node M of an input number, as appropriate for the selection, and the constant node (1) connected to the unsigned output number node N. The 1N weight is set to zero and the MN weight is set to the absolute mean magnitude value of N divided by the absolute mean magnitude value of M. The mean magnitude value is obtained by ranking the selection set records by the relevant node value, choosing the median record, (or averaging two records if both qualify as median) and calculating the difference between the median value presented and the minimum value for that node. The exponent of N is initialised to one.

Signed/unsigned number input to signed number output

A **magnitude connection** (if selected) for an input number, signed or unsigned, to a signed number the structure is modified as follows:

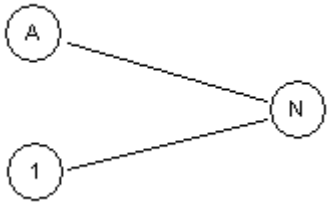
[anns012.gif]



The initialising of 1Q and MQ is as for 1N and MN in the previous diagram.

If an **absolute connection** has been selected for a signed output node the secondary output number is connected thus :

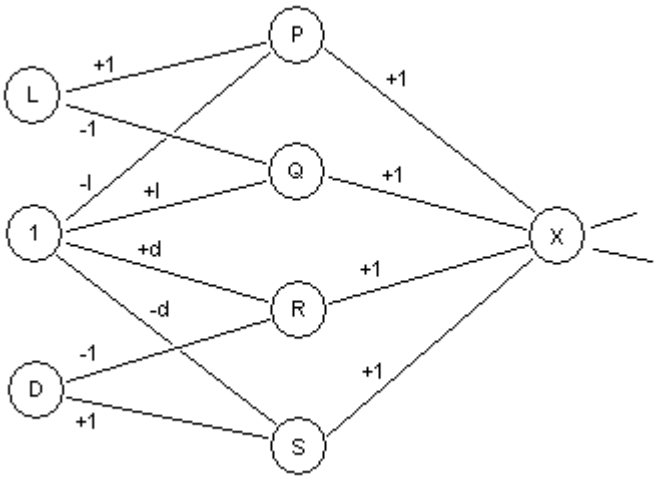
[anns013.gif]



Depending on whether the selection chosen was for proportional or inversional proportional suitability, the abs or rev-abs part of a numeric input is directly connected to the output as is the constant node. The 1N weight is set to zero and the AN weight is set to the mean magnitude value of N divided by the mean absolute value of A. The mean absolute value is obtained by ranking the selection set records by the relevant node value, and using the value obtained via choosing the median record, (or averaging two records if both qualify as median) for that node. The exponent of N is initialised to one. The output secondary sign node is treated as an independent binary singleton. The binary selection process is performed only if the absolute suitabilty indicator for the absolute selection process surpasses that of the magnitude selection process. It is then processed as a separate connection.

Additional structure for arrays

The structure for square grid arrays is the same as for singletons except for an *additional* structure :
[anns010.gif]



The above diagram shows d-coordinate and l-coordinate secondary input nodes, D and L, in a structure designed to suppress all but the selected symmetry group input. The l and d link weight values match the coordinates of the selected group. The arrangement is such that if the input does not match these values at least one of the P, Q, R ans S intermediate nodes will be activated thus activating the X intermediate node via the fixed +1 weight connecting links in turn. The X node is connected to all the relevant secondary output nodes by the fixed -g weight suppression links (not shown). The initial narrow focus is an intentional stratagem to simplify the initial training task. In the subsequent training session, the scope of the input is automatically widened if appropriate.

The structure for line arrays is similar but minus the d-coordinate node and connected links. The structure for heaps is the same as for singletons.

Derived input

Input is said to be derived when the primary data for a new net undergoing training is connected to an existing net which then produces output that can provide further input to the new net structure. Each expressed output node from the existing net acts exactly the same as an added input node. Each existing ANN has a profile for the type of acceptable input. At least for the first implementation, derived input can only be produced if there are suitable input types that match the existing ANN profile. However, it is entirely possible that many combinations of input nodes will satisfy the criteria and therefore be qualified to produce many combinations of output to be used as derived input. Without restriction the extent of the combinations could overwhelm the general computation. Automatic typing to remedy this is a line of research reserved for the future. (Automatic typing can perhaps be achieved by detecting similar input and target output profiles between different ANNs.)

The rules for matching existing ANN input nodes to new ANN input nodes are :

- Every existing ANN primary input node must have a separate corresponding new primary input node, i.e. no existing primary input node must remain unmatched. Otherwise no derived input is produced.
- An absolute requisite is that every existing ANN primary input node must be matched only with new primary input nodes with the identical semantic type, i.e. an identical semantic type must be associated with both items. Otherwise no derived input is produced.
- Preferentially existing ANN primary input nodes are matched to identical input node types. Only if this is impossible is extended matching considered.
- Preferentially signed integer input, existing or new is matched by losing the sign node. to become unsigned Similarly signed real number input can be matched. If this is impossible further matching can be considered as below.
- Preferentially signed integer input, existing or new, is matched to signed real number input. If this is impossible further matching can be considered as below.
- Integer and real number input, signed or unsigned, existing ANN or new, can be matched by treating each as unsigned real number input. The secondary sign node is suppressed.
- Heap topology type input can be matched to line or grid by ignoring certain secondary nodes.
- If the existing ANN has only singleton input and output its singleton input can be matched against array input to produce a derived array output.

When more than one new ANN input node can be matched to existing ANN input nodes, every logical combination can be matched to provide a permutation of derived input. The secondary nodes including rev-sign, rev-abs etc. are automatically created.

The ANNS expansion phase

As indicated in the overview of the ANNS process control, when a specific structure has been trained to the extent where it is deemed unlikely to improve further, i.e. the active training phase is completed, the structure is then used as the root for a contraction and an expansion.

A set of optimal nets is kept each with a unique combination of the number of constant node links within the number of other links in the (therefore unique) net structure. The combination of other links qualified by additional constant node links is known as the **effective link count**. Normally the expanded structure, a **trial net**, is subjected to a complete active training phase to verify the expansion. If the score of the trained trial net is higher than that of any optimal net with a lower or equal effective link count, the new optimal structure is retained with expansion and contraction flags cleared (to indicate both phases are yet to be performed). Then any optimal nets with an equal or higher effective link count are jettisoned. However, if the trained trial net scores more poorly than the existing optimal nets with equal or lesser effective link count, then the trial net is jettisoned, and from the nets with a contraction flag clear or failing that from the nets with an expansion net clear, the one with the least effective link count is chosen for the appropriate action (i.e. contraction prefered to expansion).

The desired expansion for the trial net is minimal so as to discourage the memory effect during training. The expansion is undertaken in up to two sessions. The first session attempts to improve the current optimal net score by either adding a link or by splitting a “conflicted node”. If no candidate is found or if after finding a candidate, the active training phase undertaken is unsuccessful, a second session is attempted where a definite isolated improvement is made for one target for one record, but in such a way as to encourage widening of the scope under general training. Even if the expansion remains relevant to an isolated incidence, this may allow further progress for other targets and other records, especially if the incidence “remembered” is actually incorrect data.

1st session expansion

A selection set is determined as described in the “ANNS initial bedding phase” section. The selection set records are expressed for the current working net. Each intermediate and output node is given a **potential improvement indicator** value according to the number of record instances totalled for records where the node in question is active (i.e. one or more superior link weights are active) but the input quantity does not match the target quantity. The nodes are then processed in the order, random within descending potential improvement indicator value. A **node access sequence number** is allocated according to the reverse order, i.e the first node in sequence has the highest number and the last node access sequence number is set to one.

A **selection improvement indicator** is calculated for the node in progress. This value reflects the number of record instances that might be improved but divided by the notional number of links that need to be added. If the node is the first to be processed, the selection improvement indicator initialises the best selection improvement indicator, and the node is marked as the best candidate for expansion so far. Otherwise, the ratio of the square of the best selection improvement indicator to the square of the current selection improvement indicator is used as a factor in a formula to potentially select a new best node. The other factor used is the ratio of the node access sequence numbers – this compensates for the unequal number of participations for the nodes in the selection process. The resulting product of the two factors for each of the the two nodes under consideration then form the weighting for a random selection. The random element gives a useful degree of variance and reflects the unavoidable approximate nature of the indicators used.

If a node is selected for *potential* expansion, an **improvement comparison indicator** is calculated (more later). The square of the selected node’s improvement comparison indicators is used along with the square of the next (in sequence) node’s potential improvement indicator, as weighting in a random selection to determine if the process can be truncated at the current node. If truncation is chosen, the current selected node is then expanded appropriately and a training session is undertaken. However, if truncation is not chosen, the next node is processed using the same truncation test (but applying the appropriate potential improvement indicator), and so on until either all the nodes have been processed or, more likely, truncation is chosen.

The improvement comparison indicator is calculated by dividing the number of record intances that can be potentially improved, by a notional number of extra links depending on the type of expansion. On the grounds of eliminating improvement only to a single record of the selection set, an expansion selection is rejected if the improvement comparison indicator is not equal or greater than twice the average number of record instances per selection set record.

There are three categories of expansion examined in a first session :

1. Linking a **target node** to the constant node.

2. Linking a target node to a single **donor node**.
3. Conflict separation applied to separate one from a multiple of subordinate links that cause target conflict.

A **donor node** is a secondary node from which content can advantageously be drawn to supply the **target node**. A donor node can be input or derived input or intermediate or even output, and potentially can introduce loops. A target node is any intermediate or output secondary node that presents an ideal target range with input differring from at least two records of the selection set.

For each expansion case examined, a **connection complexity quantity** is determined along with a **target improvement count** from which the **improvement comparison indicator** is calculated. The connection complexity quantity notionally reflects the number of links needed to connect the donor to the target link. The target improvement count is the number of record instances within the selection set which (perhaps notionally) can potentially be improved upon. The improvement comparison indicator is calculated by dividing the target improvement count by the connection complexity quantity.

Up to three types of improvement comparison indicators can be produced for each node examined - a **target node to donor constant node**, a (other than constant) **donor link**, and a **conflict separation** type. In order to find the improvement comparison indicator for the node, a choice is made between the three types of indicator by using the squares of each indicator, if present, to form the weighting for a random choice.

1st session : Donor link examination – singleton non-binary target

For the current target node, a **positive match string** and a **negative match string** is created. Where i equals the number of record instances for the specific element's selection set record, each element of each string is assigned +i, -i or 0 corresponding to an improvement, degradation or neutral result when applying the cadidate donor notional content, zero or a very small postive or negative quantity according to the string, to the input to the target node. (In the wider scheme, the quantity is applied to the imaginary ling as an imaginary weight, and if later the update is selected for expansion, it is known as the **expansion weight**.) Zero donor content always produces a neutral zero element. Other assignments depend on which type of target range the current input to the target node falls.

If, before application, the input quantity is within the tolerance target range but outside the extended target range (and thus the ideal target range too), then after application (adding or subtracting the notional small amount according to string type) moving into the extended or ideal target range is rewarded with a +i element, moving outside the tolerance range results in –i being assigned, otherwise 0 is assigned.

If, before application, the input quantity is within the extended target range but outside the ideal target range, then after application moving *towards* the the ideal target is rewarded with +i, and moving away from the ideal target range results in –i assigned.

If, before application, the input quantity is within the ideal target range, then after application moving outside the ideal target results in –i being assigned. If, before application, the input quantity is equal to the single ideal target value, then –i is assigned. Otherwise the element is assigned zero.

For any one target node, separately each of the positive and negative match string, is examined to see if it can potentially be better than the previous candidates examined, and for the first time if it has less than two improvement elements. If there is potential improvement, then (via pointer tags) the string is sorted into ascending donor content order.

For each string examined, a **connection complexity quantity** is determined and a **expansion suitability indicator** is calculated. The connection complexity quantity notionally reflects the number of links needed to connect the donor to the target link. The expansion suitablity indicator is calculated to be the best score possible of contiguous elements, which must be greater than one element regardless of the number of record instances, divided by the connection complexity quantity. The selection is skipped if the expansion suitability indicator is less than twice the average record instance for the selection set.

For example consider a non-binary donor node at the immediately superior level to the non-binary target node and a current "best" suitability indicator of 2.5. Say the match string before sorting was 20 elements, each with one record instance, thus :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	6	6	6	6	7	8	9	10	10	11	11	11	14	16	17	18	20	22	22
-1	0	+1	-1	+1	-1	-1	-1	+1	+1	0	-1	+1	-1	0	-1	+1	+1	-1	-1
-1	+2				-1	0	-1	+2		+1			-1	0	-1	+1	+1	-2		
-1	+2				-2			+3					-2			+2		-2		
-1	+3												-2			+2		-2		
-1	+3																	-2		

The first row indicates selection record position after sorting. The second row shows the donor node content. The third row shows the string element assignment. Elements that equate to the same value donor content are merged to form **cells** as shown in the fourth row. Then positive or zero cells that are adjacent are merged. Then negative and any remaining zero cells are merged. The result is shown in fifth row. Then the process is one of selecting, at random, triple cells which have a positive central cell equal or greater to the adjacent negative cells. See row six. This is repeated until no further triplet cells can be found. See the seventh row. The next step is to calculate the expansion suitability quantity Q for the whole string and a further three substrings. The whole string Q is found by summing the element assignment quantities, in this case –1 +3 –2 or zero. This is less than the minimum acceptable value, here 2 (and less than the current indicator of 2.5) and so is rejected. Next the "leading" substring of the first positive cell from the left and combined with any (negative) cell to the left of it is examined. Here Q is (-1 +3) / 2, the divisor which is the connection complexity factor of 2 reflects the extra link needed. Similarly the "trailing substring" comprised of the first positive cell from the right combined with any cell to its right is processed. Here Q is (+3 –2) / 2. Finally the largest quantity cell isolated from the start and end of the string is processed. Here Q is +3 / 3, where the divisor, 3, reflects the added link needed to connect a middle substring.

Note that the suitability indicator Q is halved in the special case when the donor is already connected to the target node plus the whole string is selected. This in order to reflect the added complexity of an additional intermediate node with links.

In this example none of the substrings is equal or greater than the minimum of 2 and therefore of course is not equal or greater than the current suitability indicator. If one were, then it might be chosen to supersede the current chosen donor with a degree of chance governed by the relative sizes of each suitability indicator modified by whether or not a loop would be caused and also modified by the difference in structural levels. There is no precise justification as to how to calculate the weightings which arbitrariness is partly the reason for a large degree of randomness in the selection. However if the suitability of the new candidate donor is equal or greater than the encumbant candidate donor then a random choice is made but weighted by the two quantities for encumbant and new candidate node as follows:

$$W = Q^2LI$$

where

Q = the candidate's suitability indicator

L = 0.25 if the candidate link would introduce a loop (detected by following the subordinate links) else L = 1.

I = 0.5 if, when connected to the candidate link, there is more than one path back to the donor node else I = 1.

The chance of the new candidate being chosen is $W_n / (W_e + W_n)$

where W_n is the weight calculated for the new candidate and W_e is the weight calculated for the encumbant.

The justification for reducing the weighting when multiple paths are present is to put evolutionary pressure on developing existing paths whilst nor ruling out supplementary and possibly replacement paths from eventually forming. Loops are likewise discouraged to emphasise the development, in the early phases, of node exponent quantities.

1st session : Target node to donor constant node examination – singleton non-binary target

If the target node already has a superior link to the constant node, or if the target node is of the binary type, then this examination is skipped and examination of the next expansion type is embarked upon. Otherwise, this is a simplification of the previous examination type. A positive and negative match string are prepared as before. These are individually summed and Q, the candidate's suitability indicator is assigned the larger quantity. No selection is made if Q is less than twice the average number of record intances per selection set record. Otherwise the selection weight, W_n, is set equal to Q for the randomised selection.

1st session : Singleton binary target

If the target node is binary, unlike for non-binary nodes, a small alteration to the input quantity cannot have any intermediate effect so it is necessary to consider a range of candidate expansion weights, not just a small positive or negative quantity. The candidate expansion weights are those calculated to match the targets where the selection record gives an imperfect result. This set of (unique) values is applied to the entire selection set to produce a number of **expansion match strings**. The method is then the same as previously described, but taking into account the increased number of match strings. The candidate expansion weight is noted for application to the structure expansion if the target to constant link is finally selected.

1st session : Conflict separation

Intermediate nodes that have more than one subordinate link are considered for conflict separation, where the node and its superior links are duplicated but one selected subordinate link is separated from the rest so as to allow independent back propagation. A subordinate link is said to be in conflict if its weight has an ideal target range which does not coincide or overlap the ideal target range of another of the same intermediate node's subordinate links. Each node/subordinate link can form a separate selection. Per subordinate link, the number of record instances is summed for all the records in the selection set. These totals are divided by the number of superior links to provide the expansion suitability indicators. The squares of the expansion suitability indicators are used in the usual fashion to determine weightings when selecting between competing expansions. Note that the conflict separation merely separates a single subordinate link from the others, relying on the contraction process to group subordinate links on the basis of reducing the number of links in the net.

1st session connections

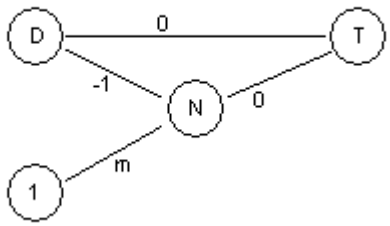
If the first session produces an expansion, then the appropriate connections are made and a training session is embarked upon. The connection scheme is as follows :

Connecting a constant node

A link is made between the constant node and the target node. If the target node is binary, the chosen candidate expansion weight is used for the link. Otherwise the link weight is initalised to zero.

Connecting a donor node

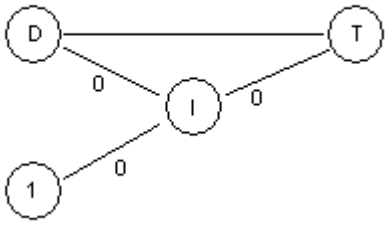
For a full range connection where the donor node is not already connected to the target node (including input nodes) : [anns023.GIF]



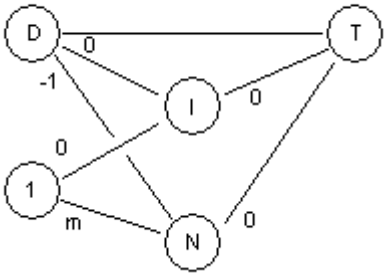
This allows for either a proportional or inversely proportional relationship to be developed between donor and target under training, the unused part being eliminated upon a contraction phase. The weight m is set to the maximum value of the donor content, either as specified for an input node, or in the case of an intermediate node as found by expressing the entire load set to find the maximum content. (This produces an intermediate reverse node N).Additionally, if no link between the constant node and the target node exists, a zero weight link is inserted (not shown).

Note that logically, for donor input nodes, there is no need to create an intermediate reverse node. However, it is simpler to do so – the extra links are removed at the next contraction phase.Instead, links with weights initialised to zero, are made directly to all the secondary nodes.

If a donor secondary node is already connected to the target node, then a futher intermediate node must be created : [anns024.GIF]

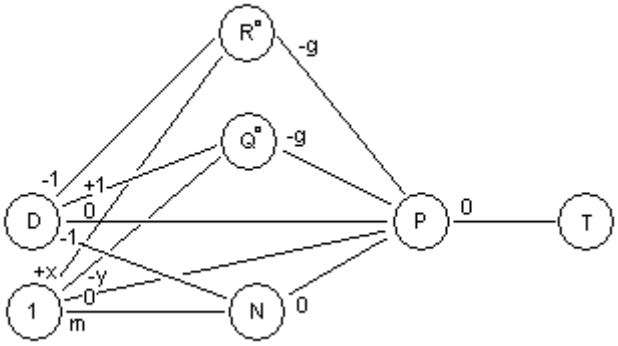


The intermediate node I is connected so as to provide an alternative route to the DT link. This structure is used only where otherwise two links would be introduced between the same two nodes. An additional intermediate reverse node is created as previously described :
[anns025.GIF]



Where a substring is selected additional structure is inserted to replace the donor node with a modified intermediate node, the detail structure depending on the type of string truncation. The modified intermediate node is then treated as the donor node would have been. The following diagram shows the scheme for both beginning and end truncation.

[anns032.GIF]

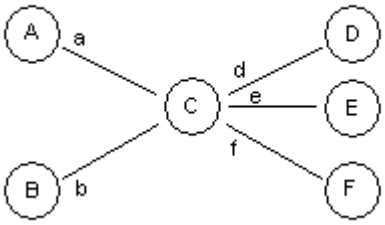


The intermediate R and Q nodes perform suppression for below the donor beginning value, x and above the donor end value, y, to select the desired string. The x value is calculated as halfway between the donor value of the first record in the string and the preceding donor value *in the entire load set*. The y value is calculated as halfway between the donor value of the last record in the string and the next donor value in sequence *in the entire load set*. The node exponents for R and Q are fixed at zero. The weights for the links RP and QP are fixed at the notional large negative number.
(Note that any input value to a –g weight ensures that the subordinate node node is supressed regardless of other input.)

If the donor node is already connected to the target, there is no need to introduce any other intermediate node because the P node does that job. If the selected string is not truncated at the beginning, then R node and its links are omitted. If the selected string is not truncated at the end, then the Q node and its links are omitted.

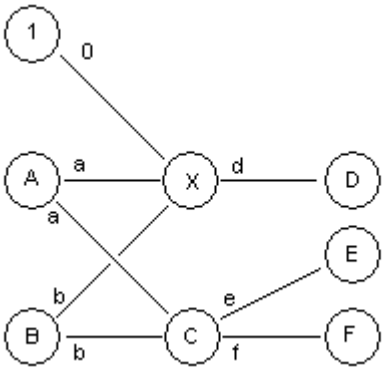
Conflict resolution connections

[anns030.GIF]



Say in the above diagram the link selected as “in conflict” was CD. Then, by cloning the C node into the new X node (i.e copying the node components and input links), this selected link would be hived off as in the next diagram :

[anns031.GIF]



Note that if the original node was not already connected to the constant node then a zero weight link is introduced from the constant node to the cloned node.

Extending the method to cope with arrays

In general the method for treating record data for array elements is too pool the elements which are symmetrical. The match string scores are the accumulation of the scores for each package of symmetrical elements. Heap array elements are deemed to be just one package of symmetrical element so the match string score for each record is an accumulation of the scores for each individual element. For line arrays pairs of elements sharing the same value l-coodinates are pooled. Similarily for square grids, elements sharing the same values for the combination of l-cooordinate and d-coordinate are pooled. If selected, extra connectivity to select the appropriate l-coordinate and d-coordinate element is made. Because the array structure is accepted as a “natural given”, these extra connections are not taken into account when calculating the connection complexity quantity. If the selected extra connections are made to select out the chosen package of elements, at this stage no attempt is made to widen the selection. This is left up to the normal training process in due course.

Note that only links between a subordinate array node and a superior node that is either a matching array node or a singleton node are legal. Illegal connections are of course not considered.

For details of additional linkage in order to isolate the selected symmetrical elements of an array see “Additional structure for arrays” within “The ANNS initial bedding phase” section.

2nd session expansion

If the first session resulted in an expansion selection, a training cycle is undertaken with the modified net. If that training session fails to provid a new optimal net, or if no expansion is selected for the first session, then a second session is undetaken abandoning any expanded net from the first session. The purpose of this secondary process is to select one output node in need of target improvement for one record instance and connect with minimal structure such that only that one record instance is effected. Effectively the net is structured to "remember" this one instance. This is not in the spirit of finding principles but does have the merit of temporarily removing awkward and maybe invalid data enabling progress with the rest of the data. The selection of the output node and record instance is done in such a way as to minimise the additional links needed. The additional structure may be readsorbed in a subsequent contraction phase.

The detail logic is as follows:

STEP1
Examine records that uniquely trigger each existing memory contributer node. For each such record determine if the contributor nodes can be connected to secondary output nodes not yet connected to it where the output node and also requires a target update. If more than one such instance is found, select one instance at random. If no such instance is found got to the next step. Otherwise connect the selected output node to the existing C node as detailed in the diagrams below. Thus in cases of multiple output nodes, "memorising" may be progressively applied at each successive expansion phase.

STEP2
Set the set of records for examination to the entire load set. Set the list of candidate donor nodes to all input nodes and all derived input nodes. Note that intermediate nodes are not included because these will tend to change when trained and thus invalidate the one value structure.

Select a secondary output node randomly within the most number of target update values, thus effectively temporarily removing a target which interferes with others (unless only single instances of target updates remain).

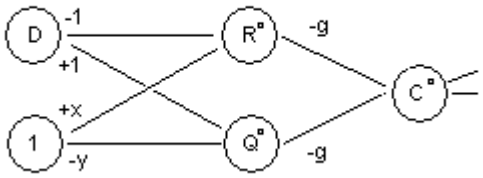
STEP4
Select a "focus" record at random that produces a target update in the chosen output node. Reduce the set of candidate donor nodes to those that have a greater than zero output value for the focus record.

STEP 5
For each node in the selected set, determine the number of output values that duplicate the node's output value for the selected record throughout the set of records listed for examination.

STEP 6
Select out one node from the selected set with the fewest number of duplicates choosing randomly between nodes possessing the same number of duplicates. Add node to list of selected donor nodes. Reduce the set of records for examination to those that contain the duplicated selected node value. Remove the selected node from list of candidate donor nodes.

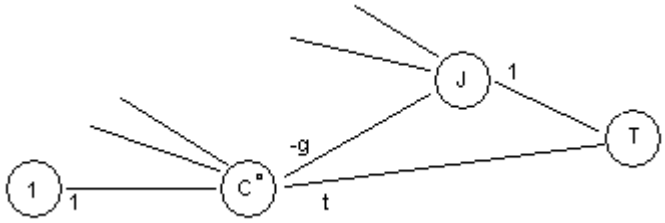
STEP 7
If the list of records for examination is not null go to STEP 4. Otherwise connect the selected donor nodes to the suppression structure for the node and output value selected.

For each donor node a pair of suppressor nodes are created and connected to a memory contributor node, C, as shown below :
[anns034.GIF]



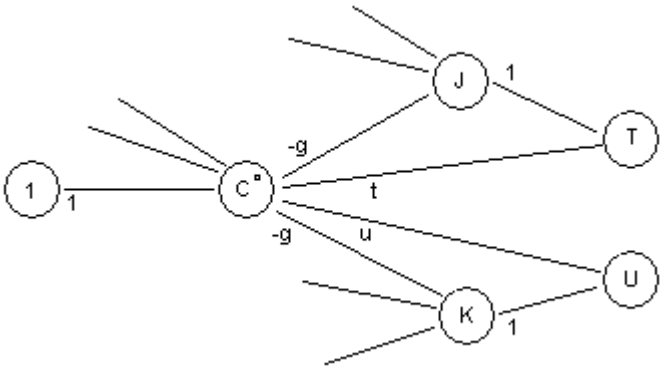
The intermediate R and Q nodes perform suppression for below the donor beginning value, x and above the donor end value, y, to select the desired string. The x value is calculated as halfway between the donor value of the first record in the string and the preceding donor value *in the entire load set*. The y value is calculated as halfway between the donor value of the last record in the string and the next donor value in sequence *in the entire load set*. The node exponents for R and Q are fixed at zero. The weights for the links RC and QC are fixed at the notional large negative number.

Having connected all the donor nodes to the contributor node, the contributor node is connected in the following manner :
[anns035.GIF]



The target node is duplicated as an intermediate node J, i.e. the T superior links are replicated for J. J is connected to the original target node with a link weight of 1. The constant node is connected to the contributor node in order to supply positive output when the contributor node is not suppressed. The combination of a zero exponent, a suppression link CJ to knock out previous contribution to the output node, and a CT link weight set to the target value gives the required value to the output node for the one selected instance.

When , for the same selected record, additional secondary output nodes are connected, the existing C is used. In the diagram below the new ouput node, U, is duplicated with intermediate node, K. The CK suppression link and a link weight of u is used to give the required output value.
[anns036.GIF]



The ANNS contraction phase

The purpose of the contraction phase is to produce an optimal net with fewer links than the current optimal net (where "current" refers to the net given to the contraction phase software routine by a control routine). By reducing the number of links, the training will tend to produce more general solutions than otherwise.

There are two categories of link reduction. The first is when a single link (not connected to the constant node) from a set of more than one superior links owned by the subordinate node, is removed. (This may also isolate the superior node in which case, the isolated node and its superior links are removed which in turn might isolate further nodes necessitating further reductions… and so on.) The second category is when a single node is removed along with its superior links (and potentially consequential removal of isolated nodes), and then its subordinate links are reconnected to another node (but only where this makes a saving in non-constant node links). This latter category includes the case where there exists a single subordinate link which when removed merges the superior node with the subordinate. In both categories new constant links may be added, but these do not count as significant compared with the reduction of non-constant links.

Owing to the time needed for each training cycle, and the large number of possible contractions, it would be cripplingly inefficient to try out each and every last one. Instead some form of less expensive estimation is necessary in order to limit which contraction nets are trialed. Once trialed, the net rating of each particular net configuration can be used as the basis for subsequent estimations.

Estimates of potential net ratings come in four flavours according to the combination of category of link reduction with a primary estimation method, or a secondary estimation method based on actual training attempts. For each potential contraction net, the success rate of each flavour is recorded separately, and used together with the estimate (whatever flavour) to determine the odds of a potential success. If the odds for a contraction configuration are less or equal to the number of possible combinations of contraction (depending on category type), then the contraction is deemed a candidate for trialing. (More about the prioritising of candidates for trialing later). The history of ratings and success rates is maintained for both series of contracted and expanded progeny nets. However, note that new parts of a net structure are always initialised to be subject to primary estimation giving a reasonably certainty that contraction is trialed if potentially viable.

To facilitate the above a **contraction candidate status** is maintained for each link and for each combination of node replacement.

The contraction candidate status is one of :

new	requiring primary estimation and potentially trial training – a new link, or a link sharing a subordinate node with a new link, or a new node, or previously marked as new and yet to be estimated or trialed.
primary	a primary estimation has been done but not trialed.
trialed	both a primary estimation and a trial training cycle have been done.

Overall contraction control

The first step of the overall process is to form a list of all the contraction candidate configurations. This includes an item for every link and every combination of intermediate node with every other node, but excluding any configurations that have the same non-constant links and nodes as existing optimum nets.

The second step is to process all those items in the candidate list that have a contraction candidate status of 'new'. For each item processed, a primary estimate is made.

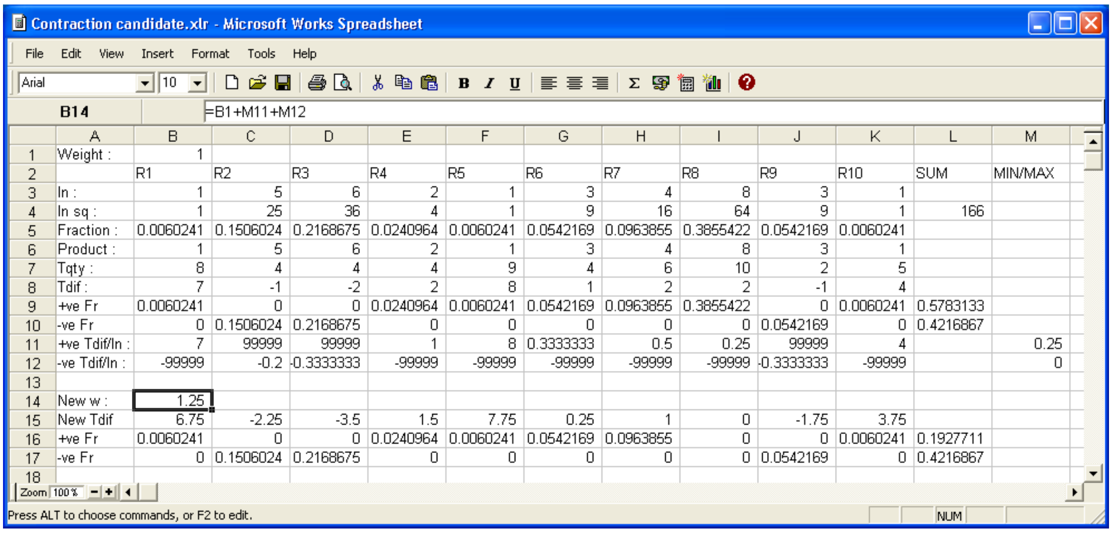
The third step is to rearrange the list into an order depending on the projected likelyhood of a successful trial, and eliminating those items that it is forecast will not be successful. If the item is 'primary' then one type of likelyhood forecast is made. If the item is 'trialed', then another type of likelyhood is additionally made and the most optimistic forecast of the two types is taken. If two or more items share the same forecast, then a random order within those items is established.

Having ordered the remaining items on the list, each item in turn is trialed until either a trial training cycle is successful or the list is exhausted.

Primary estimation – single link removal

For a single link contraction, the weights of the remaining superior links are modified to see if they can compensate for the loss.

The chosen link is temporarily removed. If no constant link exists one is temporarily created with a zero weight. Then each of the remaining links, including the constant link, is processed in a chosen random order. A partial example of the processing of one link is shown in the spreadsheet below :



Contraction candidate.xlr - Microsoft Works Spreadsheet

B14 =B1+M11+M12

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Weight :	1											
2		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	SUM	MIN/MAX
3	In :	1	5	6	2	1	3	4	8	3	1		
4	In sq :	1	25	36	4	1	9	16	64	9	1	166	
5	Fraction :	0.0060241	0.1506024	0.2168675	0.0240964	0.0060241	0.0542169	0.0963855	0.3855422	0.0542169	0.0060241		
6	Product :	1	5	6	2	1	3	4	8	3	1		
7	Tqty :	8	4	4	4	9	4	6	10	2	5		
8	Tdif :	7	-1	-2	2	8	1	2	2	-1	4		
9	+ve Fr	0.0060241	0	0	0.0240964	0.0060241	0.0542169	0.0963855	0.3855422	0	0.0060241	0.5783133	
10	-ve Fr	0	0.1506024	0.2168675	0	0	0	0	0	0.0542169	0	0.4216867	
11	+ve Tdif/In :	7	99999	99999	1	8	0.3333333	0.5	0.25	99999	4		0.25
12	-ve Tdif/In :	-99999	-0.2	-0.3333333	-99999	-99999	-99999	-99999	-99999	-0.3333333	-99999		0
13													
14	New w :	1.25											
15	New Tdif	6.75	-2.25	-3.5	1.5	7.75	0.25	1	0	-1.75	3.75		
16	+ve Fr	0.0060241	0	0	0.0240964	0.0060241	0.0542169	0.0963855	0	0	0.0060241	0.1927711	
17	-ve Fr	0	0.1506024	0.2168675	0	0	0	0	0	0.0542169	0	0.4216867	
18													

Press ALT to choose commands, or F2 to edit.

The selection set records are given a fractional rating according to the product of the square of their input qty (to the link) and the replication factor. These fractional ratings are summed for those records requiring a positive target update difference and a separate sum is made for those records requiring a negative target update difference. Those records requiring a zero target update quantity are not summed. If neither sum is greater than 0.5 then processing continues with the next remaining superior link. Otherwise the smallest quantity is calculated to add to the weight in order to satisfy at least one of the target updates but only if the relevant fractional sum previously calculated is greater than 0.5.

Having applied the correction to the weighting, the process is repeated until the fractional sum for both positive and negative target update quantities is less or equal to 0.5. [Programming note : ensure looping not possible due to rounding artifacts. In particular ensure at least one record with zero or nearest zero target update quantity is eliminated from the subsequent pass.] Eventually when neither sum is greater than 0.5, then processing continues with the next remaining superior link.

When all the superior links have been processed, as long as at least one weight has been modified, a new sweep is made (in the same random order as before). Sweeps are continued until no further progress is made.

Next, the modified net is scored as for the optimal net (using the pending set). The difference in rating (subtracting the current optimal rating from the modified) constitutes the **primary contraction estimate**.

Primary estimation – single node removal

The replaced node and its superior links are temporarily removed. Unless already existing, the subordinate links of the replaced node are reconnected to the designated replacement node. If one is not present, a constant link is added. The weights of all the added links are initialised to zero.

Using the same method as with the single link removal (see above), for each subordinate node to the replaced node, the the weights of the superior nodes are modified to compensate for the changed replacement node input. When each subordinate node's superior links have been processed, the modified net is scored as for the optimal net (using the pending set). The difference in rating (subtracting the current optimal rating from the modified) constitutes the **primary contraction estimate**.

>>
If the primary contraction estimate is positive, or more generally if the modified net rating is better than any existing optimal net with equal or more non-constant links than the modified structure, an immediate trial training cycle is undertaken, using the modified structure but reverting to the starting values of the link weights. This is to limit the effect of data exhaustion that might be caused by the estimation training which of necessity uses the larger selection set. If the

trial is unsuccessful, i.e no new optimal net is found, a **history rating defecit** for the link is calculated by subtracting the trialed net rating from the appropriate optimal net rating. Additionally, the status of the link is updated to “trialed”.

??????????????????????????????
If no successful trial training cycle has occurred, then each remaining “new” structure is trialed in primary contraction estimate order (best first).
??????????????????????????????

Trial reductions are selected according to **contraction rating estimates**. The trial reduction with the best (highest) contraction rating estimate is given priority. The exception to this is if an optimal net already exists with the trial structure, albeit with potentially different component values and constant node links. In this case the selected reduction is passed over for the next. After each type of reduction, a trial net is retrained, i.e a training expedition is undertaken. If the score is not optimal, the trial net is abandoned, but the difference between the **primary contraction rating estimate** (see later) and the actual rating is stored in a **history rating defecit** field maintained for each link or node pair depending on the type of reduction. (Note that the history rating defecit is positive if better than expected and negative if worse.) The history score defecit field is taken into account for the score reduction estimate in future contraction phases. Having abandoned the trial net, the next reduction, suggested by the order in size of the score reduction estimates, is actioned. Success of one reduction completes the current contraction phase but will normally lead to another with the new optimal net. If all non constant links and nodes have been tried unsuccessfully a further reduction attempt is made for all the constant links, i.e. trying to remove each and retrain such that the score is not reduced. Even if a constant link is deleted successfully, all the other constant links are tried in an attempt to eliminate as many as possible. The contraction phase is deemed to have failed when all reductions have been tried unsuccessfully. [Note that in a parallel computing scenario the subject optimal net can become superseded during processing. This should be detected and the subject contraction phase should thus be abandoned.]

For the purpose of estimation a selection set is chosen as for the expansion phase (see). The same set is then used in the subsequent processing of trial link reduction.

In the first category of link reduction, the trial removal of a single link aims at supplying the subordinate node input from changing the weights of the other superior links, plus modify the targets of superior node output to modify superior node and links. It is too expensive and too complicated to estimate the knock on effect of modyfying superior node targets, but for each node with superior links each non constant link is processed to estimate its potential effect on the score.

The ANNS loop optimisation phase

Loop optimisation is the process whereby the number of iterations that backward links take part in are changed and the net retrained in order to find a better net. This phase is run as a precursor to the contraction and expansion phases and after a best net is created.

No processing occurs if the current net is devoid of backwards links. Otherwise (for the first implementation), for each backwards link a random order is determined. Each backwards link is then processed in that order :

If the chosen link has more than one iteration, a random choice is made between increment and decrement (otherwise just increment). A trial net is made with the increment or decrement and a standard training phase is executed. If the net has improved the increment or decrement iteration series is continued. Otherwise, the next chosen backwards link is processed and so on.

The continuation of an increment or decrement series attempts further optimisation in the same direction until either the net does not improve, or for a decrement series, the number of iterations is reduced to one, or for a n increment series, the number of iterations reaches the implementation largest number.

Expanding the Active Set

When all the optimal nets have exhausted their expansion and contraction possibilities, attempts are made to increase the active set by taking records from the pending set. The current sets and net structures are saved for reinstatement. Complete training passes are exercised, each with varying cuts of the pending set added to the active set. Success is defined as at least one optimal net pessimistic rating being higher than that for the corresponding saved optimal net. If the process ends without success, the next level of control is called upon.

Assuming the pending set has more than four records, on the first iteration one, the initialised current **pending set expansion number**, pending set record is selected at random, and added to the active set. A training session is then done on the new set and the pessimistic rating estimate of the resulting net, using the reduced pending set, is calculated. If the rating has improved the new net replaces the saved net and training control is returned to the previuos level. If the rating has not improved, the new net and sets are abandoned, re-establishing the saved net and sets. Then the pending set expansion number is incremented, and assuming there will be at least four records left, that number of randomly selected pending set records are transferred to the active set. The process is then repeated until success or the pending set is exhausted i.e. down to just four records.

Initialising a new net structure

Whenever a net structure is newly created or changed, it is required to allocate a training step size to each and every component. Too big a size would result in unstable training ascents. Too small a size would result in unnecessarily lengthy ascents. The most important aspect of this is to obtain a reasonable size for each component relative to all the others across the whole load set (the active set will normally be too small to be adequately representative). It is anticipated that using the entire load set will not materially affect the training purity of the pending set because there is no direct link between step size and component update choice. Also note that the overall step size is not critical because this is effectively amended during training expeditions whereas individual step sizes relative to the others is not.

Each record of the entire load set is back propagated so that two values can be accumulated per component. The totals for these values are not added to where the component is either not active, or the component values lies within an incoherent ideal range (see “The ANNS back-propagation method” section for definition). If qualifying, the values accumulated for each component are :

I_c	the number of record instances
I_cQ_c	the product of the number of record instances with the absolute difference between the component input quantity and the nearest ideal range bound.

The **training span** is given by :

$$t = \Sigma I_cQ_c / \Sigma I_c$$

The training step size is calculated as one sixth the size of the training span.

Note that this can give rise to zero training steps for some components, but obviously not all components unless a perfect score is achieved.

Scoring ANNS best nets

Before applying any adjustment, the maximum score for each record is 1. However individual records may be assessed according to their replication factor, F. Usually this defaults to 1. This can multiply the score but should be considered as indicating the number of duplicate records to be used. For example, when capturing data, it can be useful to retain duplicates (exact matches of input and output values) that indicate the frequency and therefore the importance of some records and use the record replication factor to encourage the net to match those records rather than less important ones. Another use of the replication factor is to reflect rewards such as betting odds. But notionally, even for fractional replication factors, the record is handled as if it were a multiple or fraction of the subject record. Each unit's worth of a record is referred to as a record instance. Record instances are used in calculating the size of a set whereas alterations to sets are achieved by adding and removing entire records.

For a simple binary value output node net, the effectiveness of that net is represented by the percentage of hits i.e. the percentage of records that produce an exact match to the target. However for more complex nets containing multiple output nodes or containing a greater than binary value output node, a score is needed which reflects near misses as well as hits. The main purpose is to allow intermediate optimal nets to be chosen on this basis and thus evolve towards the perfect 100% score. Note that for some problems the final hit rate may be quite low but the system might be useable because the output values are accurate enough for practical purposes. The method of scoring also helps evaluation of the final best net, although the pure hit figure is quoted along with the more "fuzzy" score, so nothing is lost. However, the score encompasses a concept of "partial hits" and this allows a better estimate of error margins than would otherwise be possible.

The main problem is just how much weight to give to near misses so as to reflect a quality of partial hit. At first sight this seems an imponderable question. However, the somewhat ad hoc solution proposed, although not fully justifiable in every detail, serves the purposes it is intended for:

The maximum score for each record instance's output node is 1. Zero exponent output nodes thus score zero or 1. "Don't care" null output node content score one. Otherwise, non-zero exponent output nodes are evaluated according to how close the resulting value is to the target according to the general formula:

$$S = 1 - (d / m)^2$$

where d is the absolute deviation from the target in units, and m is the maximum magnitude of deviation possible given the output node's minimum and maximum values.

For array and indicator output nodes with x elements, each element is scored according to the above formula but then divided by x. The score for the output node is then simply the sum of the element scores.

For multiple output node nets, individual scores for each output node are multiplied together to give one overall figure e.g. 0.5 x 1.0 x 0.8 would give an overall figure of 0.4. The mathematical basis for this is one of combining probabilities. Array output node elements are treated as for individual singleton output nodes.

In the initial stages array output is likely to be filtered and links to singleton output nodes may on occasion be missing. Unlinked node elements do not contribute to the score. Note that this is very different from assuming zero input to the unlinked output node or output node element.

To score a set, each record score is multiplied by the replication factor, F, and accumulated. The rating for a set is the score divided by the maximum possible score (with don't care output scoring one). When using the reference set, this is the **best net primary rating**, but there is one more step to calculate the **best net pessimistic rating**.

A **best net pessimistic rating** is used when comparing nets for the purpose of deciding whether the improvement in a net justifies replacement of the **best net**. Except for the case when there is no external set, the best net pessimistic rating is an estimate that can be conservatively used to predict external set performance.

If there is no external set, the **best net pessimistic rating** is simply set equal to the best net primary rating. But if there is an external set, further processing is done on a selected set. In the unlikely event that the load set rating is lower than that of the reference set, the internal set is selected. In the event that the pending set rating is lower than that of the reference set, a combined pending and reference set is used. Otherwise (and usually) the reference set is selected.

The logical justification for a pessimistic rating is the supposition that the reference or other selected set is an over-optimistic selection from the combined external and selected set. In theory an attempt should be made to create a pseudo-external set with a slightly inferior rating such that an optimistic selection would statistically mirror the selected set. And having achieved this, the pseudo-external set rating would form the best net rating. However creating the

pseudo-external set involves complex calculations which are greatly affected by underlying assumptions which seem somewhat arbitrary. Thus a simplified approach is justified as long as the key behaviour in selecting upgraded best nets is achieved in practise. In particular the pessimistic rating must tend to be lower when :

- the selected set rating is lower,
- the number of records in the selected set is lower,
- the mix of replication factors in the selected set is more uneven,
- the external set is larger.

Before any calculation is done, any duplicated records in the selected set are detected and temporarily and notionally combined using the replication factor. Duplicate records are ones that produce identical output values (after rounding for the specified precision) and also input values that do not significantly differ. Input values are deemed not to be significantly different if all intermediate values also result in the same output. Since there might be many combinations of input values intermediate to a pair of candidate duplicate records, a practical compromise is to try out a large but arbitrary number of intermediate values chosen at random and if the resulting outputs are found to be the same it is assumed that the candidate pair qualify. For subsequent net evaluations, the combined records must be individually reinstated because the changes to the net will very probably result in different output values possibly giving different duplications. Duplicates will be more prevalent in the early stages of the net development. As the net becomes more complex it will tend to be more capable of discriminating between similar inputs. However, there will remain the possibility that genuine duplicates are present, especially if the data was originally captured by an automated process.

The simplified method used for the pessimistic rating calculation involves three components. First there is an element of degrading the reference set scores by a small quantity. This applies even if the reference score is perfect. Second, for selected sets with less than perfect scores, a pessimistic median is taken of a suitably large number of samples from a pseudo-external set based on multiple copies of the selected set. This reflects the effect of a mix of replication factors as well as the score distribution between records. Finally the combination of degraded and median sampled ratings is corrected for the effect of the size of external set relative to the selected set.

The calculation for the degradation factor is achieved by taking the average deviation from the scores of each incidence in the selected set when increasing or decreasing an individual output node value by one unit. Thus for a perfect score incidence, two calculations are made for a specific output node, if its value can be both incremented and decremented. Also the process is repeated to obtain deviations given for each output node if more than one is present. Each deviation calculation is factored by the replication factor. The sum of the deviation calculations is divided by the sum of the replication factors applied to each calculation, thus providing a weighted average deviation, a.

For selected sets with less than perfect scores, each pseudo-external set sample is taken by selecting the same number of records as in the selected set but individually picking them at random. (Thus most samples will have duplicate records and will miss others from the selected set.) The rating for each sample is kept. Once a requisite number of samples is taken a 25% percentile median rating, m, is found. The number of samples taken is initially 32. The median taken is simply the value for (in this case) the 8^{th} record from the samples taken in score value order. If both the average rating for the samples deviates from the selected set rating by less than the previously calculated weighted average deviation (see previous paragraph), and the 25% percentile value is less than the average rating, then the 25% percentile is accepted. Otherwise a further 32 samples are taken and the test is repeated (using all the samples taken to date) until successful. Note that convergence is mathematically guaranteed.

The estimate for the number of records in the universal and hence the number of records in the pseudo-external set (E) is, by default, derived from the number of possible combinations of input values. This default will tend to be an over-estimate but this merely leads to a more conservative pessimistic net rating. The estimate can be tightened up by using information proffered by the **net creator**, either a direct estimate of the number of records in the universal set, or individual estimates of significant values within input ranges for input nodes, or individual estimates of significant combination of values for groups of input nodes. Note that input values should ideally be pre-processed in order to present a contiguous integer range to the input node but this can be impractical so is not enforced. For practical computation purposes (to avoid overflow) a large estimate can be truncated to a standard large number without significantly harming the pessimistic net rating calculation.

The best net pessimistic rating, b is given by the formula :

$$b = r - a - ((r - m)E / (R+E))$$

where R is the number of records in the selected set, and where r is the selected set primary rating and where the other terms are as defined above.

Scoring ANNS working nets

Working nets use different selected data sets from that used for the best net calculation. Optimal nets use the pending set and active training nets use the active set. The only other difference is that at the stage of calculating each data element score, a modification can take place using the **population factor** in order to reduce the influence of common output target values for each output node singleton, array element or indicator element. The population factors are created during the first initial bedding pass to be used for the duration of the training session.

The initial calculation for each output node singleton, array element or indicator element is the same as for the best net calculation, giving a value from zero to one. If the relevant population factor is greater or equal to one, then the original score for that element ii used unchanged. Otherwise the following formula is used to calculate a modification to be subtracted from the individual score :

$$m = e\,(1 - r)\,P$$

where

- e is the original individual score,
- r is the primary score for the selected set, without modification (and thus must be calculated first)
- P is the relevant individual population factor.

Then the primary and pessimistic ratings are calculated as for the best net, but of course substituting the modified selected set rating. Pessimistic ratings need only be calculated when comparing between working nets with different selected sets.

The ANNS outer control

The outer control is concerned with the allocation of load and reference set records for each of potentially many main evolution cycles. By default the outer control also decides when there is no further point to trying a differing allocation of records and thus halts the processing. However, the user has the option to specify a maximum time and/or a maximum volume of processing and/or a sought for rating target. If the outer control halts processing before the maximum time specified, the processing is restarted from scratch (albeit retaining the best net data).

By default, the initial main cycle starts with a null net, and the internal data set divided randomly but in equal number between load and reference sets. If the user believes the internal data set to be either generous or meagre, he can chose a different proportion in order to speed up the process. If a null reference set is specified, the outer control is reduced to running the evolution process for a single cycle.

At the end of the initial cycle either the primary rating (best net score divided by number of record instances in reference set) is perfect i.e. equal to 1, or less. We will consider the simpler case of a perfect primary rating first:

The pessimistic rating is always less than the perfect primary rating, so it can only be improved by using fewer load set records. Having potentially used too many load set records, the only option is to start afresh but with a smaller load set and consequently a larger reference set. The results of the initial cycle are kept for potential further use. The desired proportion of load set to reference set size is halved. The allocation of records is again random and ignores the previous allocation. The net is set back to null and the allocation of records from the load set is again done from scratch. Note that theoretically the load set might be tiny especially if the user overrode the default to allocate very few load records. Thus an arbitrary minimum of ten load records is set and if the new target is less than this, processing is judged complete.

In the second case, the initial cycle produces a best net with a less than perfect primary rating. If the data is imperfect then the achieved primary rating may be the best possible. However, at this juncture with only one cycle to go on, it is assumed that further improvement is possible. So in this simple case, a number of records are chosen from the reference set to expand the load set so as to continue the evolution of the current net but making sure that the new initial primary rating is not higher than for the current reference set and also that it is theoretically possible to improve on the pessimistic rating too. The target for the number of records in the load set is taken from the lesser of $2R_L$ and $R_L + R_R/2$ where R_L is the current number of records in the load set and R_R is the current number of reference set records.

The transfer of the requisite number of records to make up the increase proceeds by choosing the highest scoring records from the reference set and moving them to a temporary working set. If all the records transferred have a perfect score, then one is swapped with the record with the highest imperfect score. Next a large number of potential swaps are performed by randomly selecting pairs, one record from the working set and one from the reference set, and then swapping only if the reference set primary rating would be closer to the current primary rating (i.e. for the combined working and reference sets) without overshooting i.e. becoming greater than the current primary rating. The number of iterations is set to the product of the number of records in each set - which should be enough to provide a sufficient degree of variety within the constraint of maintaining an approximate primary rating.

The next step is to check that the new reference set would be theoretically capable of improvement, i.e. a better pessimistic rating. A projected maximum pessimistic rating is calculated :

$$P_{max} = (I_R - i_H) / I_R$$

where i_H is the number of instances in the record with most instances in the reference set, and I_R is the total number of record instances in the reference set.

In the case that P_{max} is less than or equal to the current pessimistic rating, the evolution process is skipped and the history entry is made to indicate that the reference set split has plateaued. Otherwise the evolution process is continued.

For subsequent cycles we must consider the more complex general case. Each cycle results in a pessimistic rating which can be plotted against the load set size to form a graph. The results can then be classified as follows:

1. The least load set size that is plateaued (which might also be the peak pessimistic rating).
2. The peak pessimistic rating (favouring the least load set size where there are identical ratings).
3. Other than the above i.e. non-plateaued and not the (best) peak pessimistic rating.

The target load set size is limited to the gap either side of the peak result to the preceding or succeeding result. If there is no previous result the target load set size is halved. If the peak result is not plateaued and also there is no succeeding result then the target load set size is given by the lesser of $2R_L$ and $R_L + R_R/2$. If the result is plateaued then the target load set size is set to halfway in the preceding gap. Otherwise the load set size is set to halfway between either the preceding or the succeeding gap depending on the which is bigger and favouring the succeeding gap if the gaps are of equal size. When the chosen gap is only one record, since this can not be subdivided, the outer control loop is terminated (although it may be repeated from scratch if the user has specified a longer time). Where there is no preceding net, the records are allocated at random, otherwise the allocation procedure via an intermediate working set is followed.

GLOSSARY

training pass
The collection of training expeditions undertaken with the different selected current active subsets upon the same net structure is termed a **training pass**
……..

REFERENCES

AANNS Explained.odt An intelligent layman's explanation of AANNS.