GMR complexity – a preliminary analysis

The computational complexity is measured by means of flops. A flop is a floating point operation, i.e. the amount of work associated with an operation of e.g. addition, multiplication, subtraction, or division, …

The computational load for GMR is mainly in the learning and linking phase. The most time consuming tasks are distance computation. In the merging phase, the main task is to check the linking matrix and merge the objects (i.e. level 1 neurons) whenever possible, there is no distance computation involved in this stage, and we will neglect the computational load of this merging phase.

1. Learning

Assumption: 2 layer exinSNN are employed, each exinSNN has 2 epochs of learning.
Parameters:

N – number of augmented training data points
d – dimension of the augmented training data points
n1 - the number of first layer neurons is n1, n1<N
n2 - the number of second layer neurons, n1<n2<N

In an average situation, the number of flops required includes the following steps during the learning phase of GMR:

(1) 1st layer of learning:
learnF1 = (8*N*n1-2*n1^2-5*n1+6*N)*d + 4*N - 2*n1.

(2) relabelling after the first layer learning
relabelF1 = 4*d*n1*N.

(3) 2nd layer of learning:
(under the assumption of an ideal situation, each domain obtained from the first layer will have round(n2/n1) neurons during the second layer exinSNNs)
learnF2 = 8*N*n2/n1-2*n2^2/n1-5*n2+6*N)*d + 4*N - 2*n2.

Thus, the amount of arithmetic required in a typical learning phase equals to
LearnF = LearnF1+ relabelF1+learnF2
= (12*N*n1 - 2*n1^2 -5*n1 - 5*n2 + 12*N + 8*N*n2/n1 - 2*n2^2/n1)*d + 8*N - 2*n1 - 2*n2

Hence the approximate complexity of learning for GMR, which results in n1 first layer object neurons and n2 second layer final neurons, is $O(Nn_1d)$. In the worst case, the time complexity is $O(Nn_2d)$, where the data set is very unbalanced, e.g. there is only one domain that can have more than 1 second layer neurons.

2. Linking

Assumption: 2 for each presentation of an input data point, an average of k (k<n2) neurons have been counted as linking candidates.
linkF = (4*n2*d-(k-1)*6*d+(k-1))*N

Thus the time complexity for linking, which involves n2 final neurons, is $O(Nn_2d)$.

The main computational part of the linking involves the distance computation for each presentation of an input data. E.g. in order to obtain the k nearest neighbors (in the final neurons) to the input data point, $4n_2d$ flops are needed in the distance computation, though branch and bound search techniques can be used to decrease the amount of distance computation to e.g $2n_2d$, the largest order of the computation in linking is still $O(Nn_2d)$.

3. Recall

Assumption: m<d dimension of a test input is given, to obtain the corresponding d-m dimensions of output.

The computation required here is mainly the distance computation of the input to the final neurons in the limited space (dimension m<d), which can be written as 4*m*n2, is $O(n_2m)$.