CHAPTER 4
CONCURRENCY CONTROL USING LOCALLY WEIGHTED PROJECTION REGRESSION (LWPR) NETWORK

4.1 INTRODUCTION

This chapter presents a neural network concept that involves learning huge amount of data on incremental basis. It involves projection regression to find mapping of input data with output data. Vijayakumar et al, 2000, Locally Weighted Projection Regression (LWPR) is an algorithm that achieves nonlinear function approximation in high dimensional spaces even in the presence of redundant and irrelevant input dimensions. At its core, it uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space. This nonparametric local learning system

i) learns rapidly with second order learning methods based on incremental training
ii) Uses statistically sound stochastic cross validation to learn
iii) Adjusts its weighting kernels based on local information only
iv) Has a computational complexity that is linear in the number of inputs, and
v) Can deal with a large number of - possibly redundant & irrelevant – inputs.

4.2 IMPORTANT VARIABLES USED IN THE LWPR PROGRAMMING

1. n_in : number of input dimensions in the regression.
2. n_out : number of output dimensions in the regression.
3. diag_only : 1 – make the distance metric diagonal only
   0 – allow full distance metric
4. allow_meta_learning : 1 (0) – allow (disallow) second order learning.
5. meta_learning_rate : To be used if second order learning is enabled
6. penalty(γ) : Multiplication factor for the regularization penalty term in the optimization functional
7. init_all_alpha : Distance Metric learning rate initialization for gradient descent method.
8. norm : Normalization constant by which each input column is divided. This tries to make the all the inputs dimensionless.
10. max_rfs [100000] : Maximum number of local models or receptive fields (RFs) allowed.

4.3 PARAMETERS FOR HANDLING EVALUATIONS (TRAINING & TESTING) DATA SETS

The event loop

The structure of the event loop is shown in Figure 4.1. The algorithm is at one of the four action states at any given point of time. The INITIALIZE phase is used to initialize the LWPR and read in the script variables from the
After every eval_time iterations, the program goes into the EVALUATE phase where the learned model is tested against the test data set. When the number of iterations has exceeded the max_iterations count or the change of Normalized Mean Squared Error (NMSE) between the last EVALUATE phase and the current is below a THRESHOLD, the program goes into the RESULT phase. In the RESULT phase, it dumps (saves) the learned LWPR, saves the result of evaluation on the test and PAUSES (stops).

4.4 ADDING AN EXTRA PROJECTION DIMENSION

When init_n_reg=1, then, this will be a special case of one projection LWPR where the number of local projection employed by each local model is restricted to one. The distance metric adjusts in order to accommodate for this restrictions. This may take a long time to converge for regression problems in which the inherent dimensionality is high.

4.5 PRUNING AN EXISTING RF (LOCAL MODEL)

If there is a local model that elicits substantial activation in response to a training data, then, it prevents the allocation of an additional local model for that training data. Since the distance metric is changing with the gradient descent updates, there can arise cases in which there is a considerable overlap between two local models. The pruning could be due to (i) too much overlap with another one (If 2 receptive fields elicits response greater than w_prune to a training data), - one with the larger error is pruned. (ii) too high variance in the error compared to the STD of all the RFs (determined by the variable factor_prune).
4.6 MAINTAINING THE LOCAL NEAREST NEIGHBOR (NN) LIST

When using the regression analysis in applications where the input values changes smoothly it is useful to keep a neighbourhood list and perform training by looking at only the neighbouring local models which are close to each other or have a substantial overlap in their activation profiles. This saves a lot of computing resources as opposed to going through all the local models and finding out those that have enough activation to be updated. It suffices to look at the neighbourhood list to check for activations that are above the threshold and need to be updated.

4.7 SECOND ORDER UPDATES

The gradient descent updates of the distance metric is efficient if Newton’s second order gradient information (meta learning) is used. If the allow_meta_learning variable is TRUE, then the second order learning is switched on.

4.8 FORGETTING FACTOR

The forgetting factor is a variable that is used to discount the effects of the statistics computed at an earlier stage and give more weight to the recent statistics - which are a result of having experienced more data points. It can be thought of as a sliding window over which the stochastic sufficient statistics are accumulated. The forgetting factor (lambda) takes a value [0,1] where 0 corresponds to using only the current point and 1 corresponding to not `forgetting ` anything. Here, use an annealed forgetting rate which forgets more at the start (to account for unsettled learning dynamics) – specified by init_lambda and anneals towards a value
closer to one (final_ lambda) - not forgetting anything based on annealing factor tau_lambda.

Fig. 4.1 Flowchart of Locally Weighted Projection Regression
4.9 SEQUENCE OF MODULES EXECUTED WHEN A TRANSACTION REQUESTS LOCK (LWPR TRAINING) OR RELEASES LOCK (LWPR TESTING)

The step 1 to step 4 for case 1, case 2 and case 3 are executed to produce performance metrics.

1. The LWPR was trained for transaction $T_i$ on objects $(O_1, O_2, \ldots, O_n)$
2. OL testing was executed with objects $(O_1, O_2, \ldots, O_n)$ in step 2 to obtain binary value. If ‘000’ is obtained in the output layer, then the object(s) can be locked. If (001, 010, 011, 100) is obtained in output layer of OL testing, then the object(s) is under use.
3. In any case, if transaction $T_i$ is requested on object $O_i$, then OL training update weights inclusive of new patterns using the LWPR.
4. In any case, if the object is under any lock mode other than shared or no lock, then the transactions are kept under queue.
4.10 RESULTS AND DISCUSSIONS

4.10.1 FORK BASE DRAWING

Fig. 4.2 Releasing time for each object in Fork

Fig. 4.3 Locking time for each object in Fork
Fig. 4.4 Total Locking time for each transaction group in Fork

Fig. 4.5 Total Releasing time for each transaction group in Fork
4.10.2 BOLTED CONNECTION DRAWING

Fig. 4.6 Releasing time for each object in Bolted connection

Fig. 4.7 Locking time for each object in Bolted connection
Fig. 4.8 Total Locking time for each transaction group in Bolted connection

Fig. 4.9 Total Releasing time for each transaction group in Bolted connection
4.10.3 BEARING DRAWING

Fig. 4.10 Releasing time for each object in Bearing

Fig. 4.11 Locking time for each object in Bearing
Fig. 4.12 Total Locking time for each transaction group in Bearing

Fig. 4.13 Total Releasing time for each transaction group in Bearing
4.11 SUMMARY

This chapter presents implementation of LWPR for locking of transactions of object in Fork, Bolted connection and Bearing. LWPR consumes comparatively less amount of time in obtaining the final weights when compared to the time taken by FUBPA. This algorithm is very much suitable when time between transactions is very less. Chapter 5 presents implementation of Fuzzy logic for concurrency control of object in Fork, Bolted connection and Bearing.