|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | General Functions |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | decrementVal(X,Y):- |
|  | Y is X-1. |
|  | addVals(X,Y,Out):- |
|  | Out is X + Y. |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Hop Activation |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | hopHelp(X,Y,Oldo,Oldo) :- X =:= Y, !. |
|  | hopHelp(X,Y,\_,1.0) :- X > Y, !. |
|  | hopHelp(X,Y,\_,-1.0) :- X < Y, !. |
|  |  |
|  |  |
|  | hop11Activation(Net, Alpha, Oldo, Z) :- |
|  | hopHelp(Net,Alpha,Oldo,Z). |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Hop Activation all |
|  | Entire Hopfield network state, given network net activation |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | hop11ActAll([],\_,[],[]). |
|  | hop11ActAll([ H | T ], A , [ H2 | T2 ], [ N | R ]) :- |
|  | hop11Activation( H, A , H2, N), |
|  | hop11ActAll(T, A , T2, R). |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Net Unit |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  |  |
|  | netUnit([], [], 0). |
|  | netUnit([ H | T ], [ H2 | T2 ], R) :- |
|  | Unit = H \* H2, |
|  | netUnit(T, T2, New), |
|  | addVals(Unit,New,R), |
|  | !. |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Net All |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | netAll(\_, [], []). |
|  | netAll(X, [ H | T ], [ Unit | R ]) :- |
|  | netUnit(X, H, Unit), |
|  | netAll(X, T, R), |
|  | !. |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Hopfield (-1,1) training function(s) |
|  | this computes weight matrix for only one stored state |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | checkH(H, X):- |
|  | H > 0.0, |
|  | X = 0.0; |
|  | H < 0.0, |
|  | X = -0.0. |
|  |  |
|  | hopHelper( UsedHeads , [ H | []], W):- |
|  | checkH(H, Zero), |
|  | X=[ Zero ] , |
|  | append( UsedHeads , X , A ) , |
|  | createWeight( H , A , WO ), |
|  | W = [ WO ],!. |
|  |  |
|  | hopHelper(UsedHeads, [ H | T ], W):- |
|  | checkH(H, Zero), |
|  | X=[ Zero | T ] , |
|  | append( UsedHeads , X , Start ) , |
|  | append( UsedHeads ,[ H ], Heads ), |
|  | createWeight( H , Start , WO ), |
|  | hopHelper( Heads , T , Y ), |
|  | W = [ WO | Y ]. |
|  |  |
|  |  |
|  | createWeight(\_,[],[]). |
|  | createWeight( InputHead , [ H | T ], [ Unit | R ]):- |
|  | Unit is InputHead \* H, |
|  | createWeight( InputHead , T , R ), |
|  | !. |
|  |  |
|  | hopTrainAstate( S , W ) :- |
|  | hopHelper( \_ , S , W ),!. |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Hopfield weight matrix for a list of desired stored states |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | /\* add a list \*/ |
|  | addList([H | []], [H2 | []], X):- |
|  | addVals(H,H2,Y), |
|  | X = [Y],!. |
|  |  |
|  | addList([H | T], [H2 | T2], X):- |
|  | addVals( H , H2 , Y ), |
|  | addList(T, T2, Result), |
|  | X = [Y | Result],!. |
|  |  |
|  | /\* add a 2 dimensional list ie matrix \*/ |
|  | addMatrix([H | []],[H2 | []], W):- |
|  | addList( H , H2 , X ), |
|  | W = [X]. |
|  |  |
|  | addMatrix([H | T], [H2 | T2], W):- |
|  | addList( H , H2 , X ), |
|  | addMatrix( T, T2 , Y ), |
|  | W = [X | Y]. |
|  |  |
|  | hopTrain([ H | []], W ):- |
|  | hopTrainAstate( H , W ), !. |
|  |  |
|  | hopTrain([ H | T ], W ):- |
|  | hopTrainAstate( H , X), |
|  | hopTrain( T ,Y), |
|  | addMatrix( X , Y , W ), !. |
|  | hopTrain([],\_). |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Next state computation |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | /\*used for when the tail is empty. \*/ |
|  | nextState( [ H | T ],[ H2 | [] ], A , Out) :- |
|  | netUnit( [ H | T ] , H2 , Unit), |
|  | hop11Activation( Unit , A , H , Y ), |
|  | Out = [Y], |
|  | !. |
|  | /\*generates next state until tail empty \*/ |
|  | nextState([ H | T ] , [ H2 | T2] , A , Out) :- |
|  | netUnit([ H | T ], H2 , Unit ), |
|  | hop11Activation(Unit, A , H ,State), |
|  | nextState([ H | T ], T2, A, Next), |
|  | append([State],Next, Out). |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Energy |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | energy(S , W , Energy ) :- |
|  | netAll( S , W , OutAll ), |
|  | netUnit( OutAll , S , Unit ), |
|  | Energy is ((-0.5) \* Unit), |
|  | !. |
|  |  |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Update network state N iterations (N=0 is current |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | updateN(S , W , A , N , Result) :- |
|  | N > 0, |
|  | nextState(S,W, A,Out), |
|  | decrementVal(N,Count), |
|  | updateN(Out, W , A ,Count,Result), !; |
|  | Result = S. |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | findsEquilibrium succeeds if finds equilibrium in N state transition checks; |
|  | fails otherwise |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  |  |
|  | findsEquilibrium( S , W , A , Dis ) :- |
|  | Dis > 0, |
|  | decrementVal( Dis , PDis), |
|  | updateN( S , W , A , Dis , N1 ), |
|  | updateN( S , W , A , PDis, N2 ), |
|  | N1 == N2, |
|  | !. |