# Simulating Neuronal Learning during Brain-Machine Interface

**Proposed task:**
Using the Jimenez et al. article Learning to use a Brain-Machine Interface: Model,Simulation and Analysis, I will attempt to model a population of neurons in a system that learns to control a 2 dimensional cursor with increasing accuracy. I will investigate how the model changes with the variation of $\mu$ (a small positive coefficient), n (the number of neurons), and $\nu$ (the learning rate).

**Background reading:**
Jimenez, Heliot, and Carmena 2009 IEEE EMBS article

**Results:**

**Control trials:**
Figures 1-3 show the control trials. All 3 graphs were made from the exact same trial, so the errors, cursor outputs and "A" values should correspond. The alterations made in subsequent graphs, where parameters were varied, do not correspond to the exact same trial as graphs 1-3.

**Learning demonstration via decreasing error:**
Figure 1 clearly represents the learning ability of this model. As the system runs through iterative trials, (represented by an increasing learning epoch k) the error distance decreases until reaching a minimum error of 0.001. This cut off value is an arbitrary error limit set within the model. Error distance (ef) represents the distance between the cursor and the target for each trial. Using random perturbations, the model adjusts the firing rate of each neuron in the population, and analyzes the resultant error before and after the perturbation. Using the difference between the errors to alter the firing rates for the next iteration decreases the error each trial.

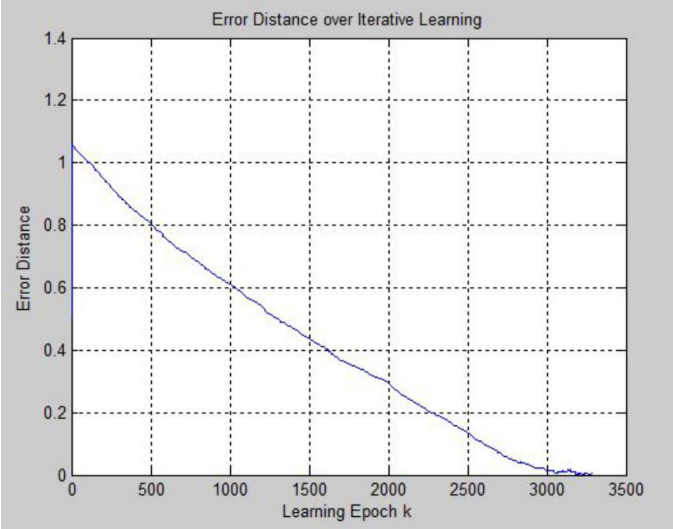


Figure 1: Error Distance Over Iterative Learning.

**Learning demonstration via cursor location:**
Figure 2 demonstrates how the system evolves by graphing the physical coordinate output Y. The first output is shown by the leftmost black circle, and over the duration of the run cycle, the output cursor location was graphed once every 50 iterations. The target is shown in red, and one can see how the system moved closer and closer to the target each iteration.

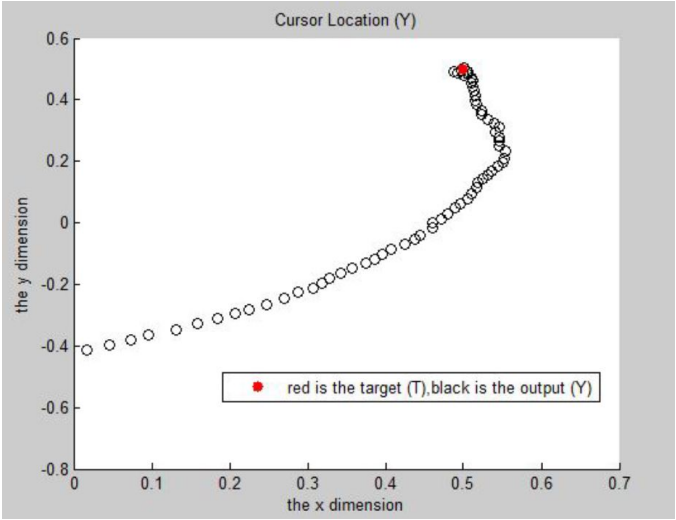


Figure 2: Demonstrating Learning via Cursor Location.

**Variations in A:**
A is a matrix of size nx2 that modulates the firing rates of each individual neuron based on the location of the target, T. The values of the A matrix will change as the number of trials increase, attempting to find the best values with which to modulate the neurons. As seen in Figure 3, the 20 values that comprise the A matrix vary as a function of the number of trials. However, these values still change significantly even when the change in error is minimal. A simple hypothesis is the values of the A matrix keep changing until the error becomes exactly zero despite how this model stops the simulation when the error falls below a threshold of 0.001.

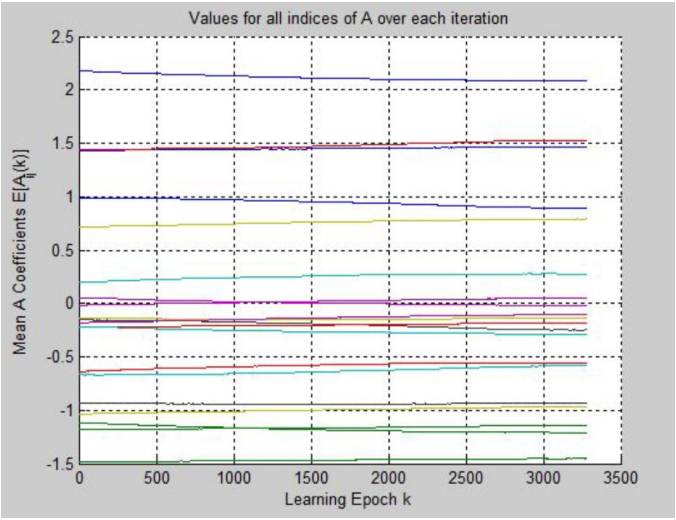


Figure 3: Values for all indices of A over each iteration.

**Variations in Model Parameters:**
Because the model works by initializing a random set of starting firing rates for each neuron, the error from the initial condition is variable. Therefore the amount of error correction that must occur, and consequently the time required to learn, is variable. Because of this a set of 15 trials were graphed (each shown as a different color) for each of the variations. With this graph it is possible to compare the averages based off the 15 trials in each of the following graphs.

**Varying size of neuron population that is undergoing learning:**
Varying the size of the neuron population did not lead to a conclusive trend. The variations in system learning were too great to make a statement on their averages changing as a function of the size of the neuron population. The results are still shown. Further testing would be required to make a conclusive statement. Jimenez et al. demonstrated the brain model using a population of n=10 neurons. It was hypothesized that as the number of neurons increases, the model should converge to a small error more quickly. This is because a larger neural network should increase the brains learning ability. This hypothesis is confirmed by Figure 4, which clearly demonstrates that as the number of neurons increases, the number of learning epochs (k) needed to reach the minimum error threshold decreases. Thus, the number of neurons and number of trials are inversely related.

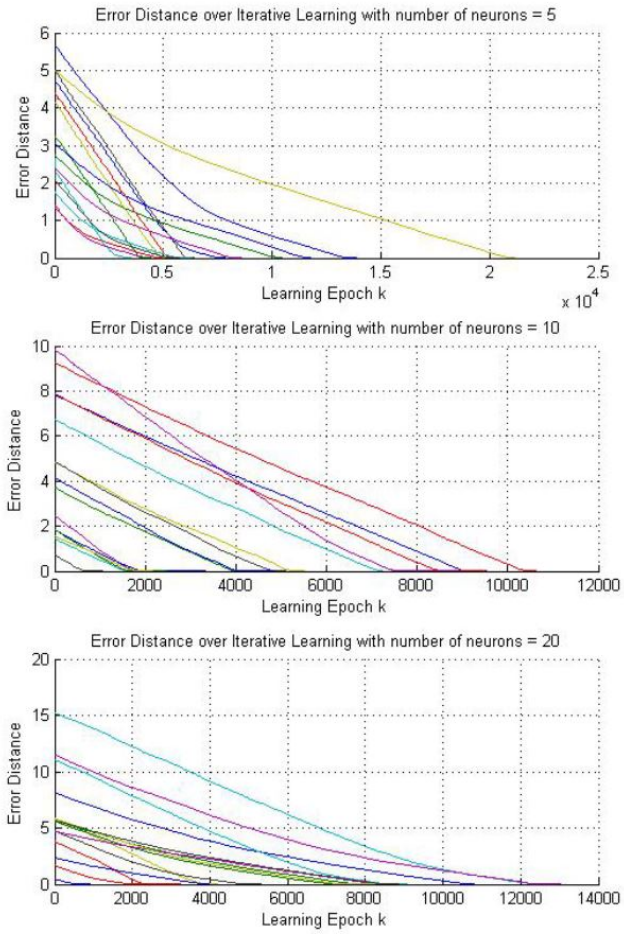


Figure 4: Measure of error distance based on varying numbers of neurons in the population while holding $\mu = 0.8$ and $n = 0.8$.

**Varying $\mu$:**

Varying $\mu$ shows a similar trend to variations in learning speed. The best tangible description for the effects of $\mu$ is a scaling factor for how drastically the system responds to a perturbation g. If the response is more drastic, the model can learn much faster than if the response is minimal. Because the system learns by changing its output in response to the effect of a perturbation g, this is a very relevant parameter in affecting the number of iterations it takes for the system to minimize error.

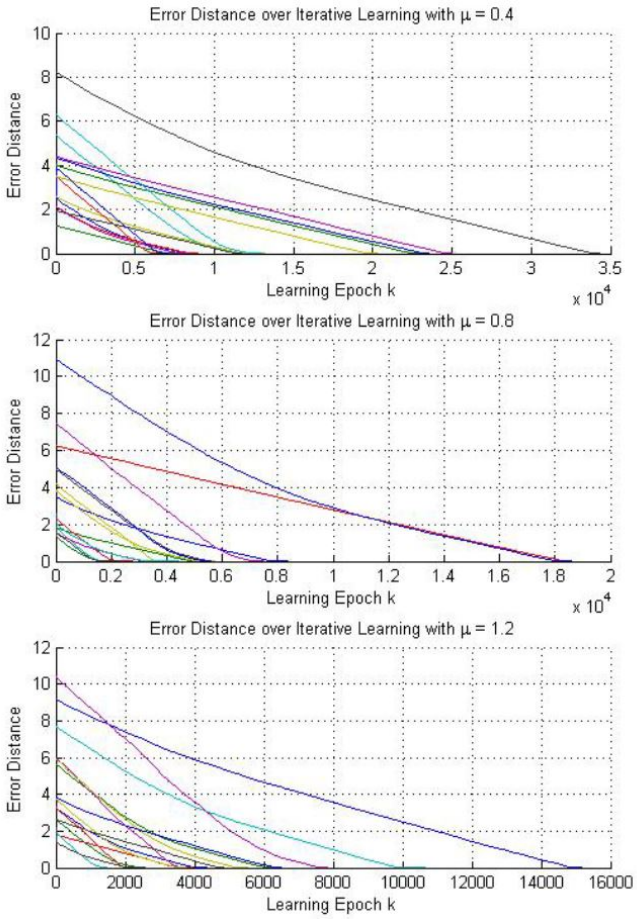


Figure 5: Measure of error distance based on varying $\mu$ (the small positive constant) while holding $\nu = 0.8$ and $n = 10$.

**Varying Learning Speed $\nu$:**
Figure 6 clearly shows that increased learning speed causes overall faster iterative learning. In other words, the error distance reaches the optimal value more quickly with a faster learning speed. Varying the learning speed (from $\frac{1}{2}$ the value posted in the article, to $\frac{3}{2}$ the value posted in the article) produced a definite trend in the number of iterations it took for the system to produce accurate cursor movement. Looking at the average of each set of 15 trials, the system was getting better at learning the task as learning speed increased. Just looking at the x axis range is a good indication of this. With similar graphical form, the smaller learning speed had a range on the order of $2.5\times10^4$, whereas the biggest learning speed had a range of around 9000. From this relatively small sample size (3 values for learning speed) a very linear trend was evident: increasing learning speed increases the speed of learning.

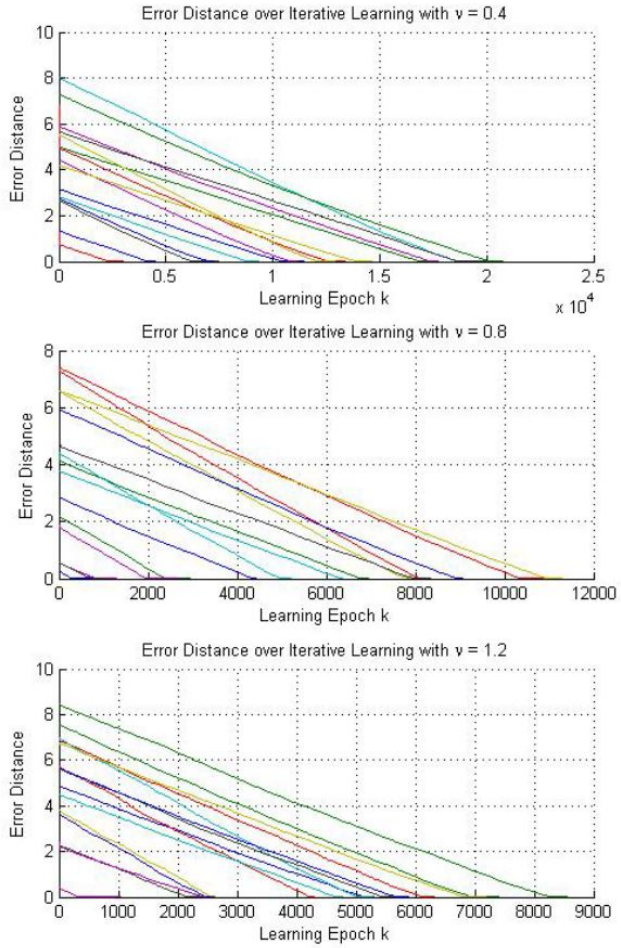


Figure 6: Measuring error distance based on varying learning speeds while holding $\mu = 0.8$ and $n = 10$.

**Supplementary Materials (MATLab Code):**

**Run Script:**

```
close all; clear all;
%%%%%%%%%%%%%%%%%%%  Model Parameters  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 10; D = randn(2,n); % # of neurons in the model
T = [ 0.5 ; 0.5 ];      % 2D target location
maxError = 0.001;       % max acceptable error (dist between target and output cursor location)
v = 0.8;                % the speed of learning
u = 0.8;                % small positive constant

%%%%%%%%%%%%   Simulation and Plotting Cursor Location  %%%%%%%%%%%%%%%%%%%
figure;
hold all;
f = randn(n,1);         % initialize a matrix of firing rates for the neurons
A = randn(n,2);
b = randn(n,1);

counter = 1;            % serves as an iteration counter at times
AValues=zeros(20,1);    % stores the value of A for each iteration so it can be graphed later

scatter(T(1),T(2),'r','filled');    %plot the target
Y = D*f;
ef = errorDistance(Y,T);
while (ef >= maxError)              %continue until the error has been minimized
    g=0.01*randn(n,1);
    ftemp=f+g;
    Y=D*f;
    Ytemp=D*ftemp;
    %animate the progress of the learning
    if mod(counter,50)==0
        scatter(Y(1),Y(2),'k');
        pause(.1);
    end
    ef(counter)=errorDistance(Y,T);
    efg=errorDistance(Ytemp,T);
    df=-u*(efg-ef(counter))*g;
    [A,b]=updateAb(A,b,v,df,T);
    x=1;
    for p=1:10
        for k=1:2
            AValues(x,counter)=A(p,k);
            x=x+1;
        end
    end
    f=A*T+b;
    counter=counter+1;
end
scatter(T(1),T(2),'r','filled');        %incase the target was covered
title('Cursor Location (Y)');
xlabel('the x dimension'); ylabel('the y dimension');
legend('red is the target (T),black is the output (Y)');

%%%%%%%%%%%%%%%%%%%%%%%%  Plotting Error  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
plot(1:length(ef),ef); grid on;
title(['Error Distance over Iterative Learning with \nu = ',num2str(v)]);
xlabel('Learning Epoch k'); ylabel('Error Distance');

%%%%%%%%%%%%%%%%%%%%%%%%   Variations in A   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;hold all;
for i=1:20
    plot(1:(counter-1),AValues(i,:));
end
title('Values for all indices of A over each iteration');
xlabel('Learning Epoch k'); ylabel('Mean A Coefficients E[A_i _j(k)]'); grid on;
```

**errorDistance Function:**

```
% UNTITLED4 Summary of this function goes here
% Detailed explanation goes here
x=T(1)-Y(1);
8
y=T(2)-Y(2);
e=sqrt(x^2+y^2);
end
```

**updateAb Function:**

```
function [ A,b ] = updateAb( oldA,oldb,v,df,T )
% UNTITLED2 Summary of this function goes here
% Detailed explanation goes here
Tmag = sqrt(T (1)^2+T (2)^2);
A=oldA+v*df*T'/Tmag;
b=oldb+v*df;
end
```

**References:**

Jimenez, Jessica, Rodolphe Heliot, and Jose M. Carmena. "Learning to Use a Brain-Machine Interface: Model, Simulation and Analysis." IEEE (2009): n. pag. Web.