Neural Networks
Lab and Discussion Questions

Discussion Questions. Some of these questions you will answer by programming and performing the computer simulations, and others you will answer by reading the articles.

1. Neurons send a spike of electrical activity down their axon, and the intensity of this electrical activity does not vary from spike to spike. Neural Network activations, on the other hand, have varied activation outputs. Does thins make neural networks a bad model of neural activity? why?
2. Describe backpropagation
3. What are the most important differences between supervised learning and unsupervised learning?
4. In what domain does Ratcliff (1990) focus on? Do you think that his conclusions can be extended to other domains? Why?
5. In the McCloskey paper, what is the distinction that he makes between simulating a phenomenon and explaining it?
6. What are McCloskey's conclusions? Where does connectionism fit within Cognitive Science?
7. Include the required pieces of code as described below. Include in your report the parts that are in bold font
8. Did you get the same results as Ratcliff?
9. What were the differences between the obtained outputs and the desired outputs during the test? What do they mean?
10. What role does the connectionist network play in Kintsch's Construction-Integration Model? Why is a neural network particularly useful for such a role?
11. How does the Rumelhart & McClelland approach to modeling learning of the past tense of English verbs differ from that of Pinker? What are the fundamental areas of disagreement?
12. What are the strengths and weaknesses of each approach? Is there any data that can be modeled well by one approach but not the other?

Lab Instructions:

We will program the model used by Ratcliff (1990) :

Let's set up the elements: We need 3 vectors that represent the activation for the input, hidden and output layers.

This is how you would create the input layer; `array(dim=4)` means that to begin, `input` will have four elements

```
> input=array(dim=4)
```

Now create `hidden` (an array of three elements) and `output` (an array of four elements).

To check if everything is OK, type

```
> input
```

then

```
> hidden
```

then

```
> output
```

and you should see something like this:

```
> input
[1] NA NA NA NA
> hidden
[1] NA NA NA
```

```
> output
[1] NA NA NA NA
```

The `NAs` mean that there are no values yet. Remember, these will be the activation levels (see Equation 2 in Ratcliff's paper)

These 4 matrices will store the deltaws (Eq 4)

```
oho=dho=matrix(rep(0,12),ncol=4,nrow=3)
oih=dih=matrix(rep(0,12),ncol=3,nrow=4)
```

Lets now create the arrays for the net_i in Equation 1. we need them only for the hidden and output layers.

```
> hnet=hidden
> onet=output
```

Now, lets create the wights as a matrix.
The weights of connections between input and hidden are a matrix with 3 cols, and 4 rows, while the weights between hidden and output are 4 columns and 3 rows.

The values are initialized to random numbers (see page 288)

for the input to hidden:

```
> wih=matrix(runif(12,-.3,.3),ncol=3,nrow=4)
```

now you create the `who` matrix. Remember, 4 columns and 3 rows.

If you type `wih` and `who`, you should get something like this (exact numbers will be different):

```
> wih
    [,1] [,2] [,3]
[1,] -0.01856777 -0.1557202 0.243149679
[2,] -0.17670973  0.2116570 0.232375807
[3,] -0.18569300 -0.1540929 0.004149762
[4,]  0.13590547 -0.1493528 0.079594320
> who
    [,1] [,2] [,3] [,4]
[1,] -0.01319009 0.25589737  0.06561314 0.0295019
[2,]  0.17116534 0.18954567 -0.18369738 0.2215547
[3,] -0.11636154 0.04034643  0.16414602 0.2301859
```

Great! now we have all the arrays and matrices that we need.

As you know we use a sigmoid function for the activation. Type the following command so that you can see it (don't save it in the source file, though)

```
plot(1/(1+exp(-seq(-10,10,.1))))
```

To compute the net_i Input to the hidden layer, and then the activation of the hidden units we use the following code:

```
for (i in 1:3){
  hnet[i]=sum(wih[,i]*input)
   hidden[i]=1/(1+exp(-hnet[i]))
}
```

How would you write the code to compute the neti input (onet) to the output layer and the activation oi (output)? Add it to the source file.

Let's now work on the backpropagation algorithm. The error signal at output is Equation 3. In R we first create the arrays to store the error signals for the output and hidden layers. We initialize them with values of zero:

```
deltao=c(0, 0, 0, 0)
deltah=c(0, 0, 0)
```

Equation 3 is

```
deltao=(teach-output)*output*(1-output)
```

With this we update the weights between hidden and output (equation 4)

```
  lr=4
  for(i in 1:4){
       for(j in 1:3){
            dho[j,i]=lr*deltao[i]*hidden[j]
            who[j,i]=who[j,i]+dho[j,i]+oho[j,i]*.5
            oho[j,i]=dho[j,i]
       }
  }
```

The dho is the Deltaw, and the oho is the old Deltaw (we use this for the momentum)
Then equation 5 becomes

```
  for (i in 1:3){
      deltah[i]=hidden[i]*(1-hidden[i])*sum(deltao*who[i,])
  }
```

Now write the code to use the deltah to update the wights between input and hidden
The vectors to be learned are 1,0,0,0 0,1,0,0 0,0,1,0 0,0,0,1. And then we will also test 1,1,1,1 and 0,1,1,0 (See table 1 in Ratcliff's paper)

In R:

```
stimuli=rbind(c(1,0,0,0),c(0,1,0,0),c(0,0,1,0),c(0,0,0,1),c(1,1,1,1),c(0,1,1,0))
```

Last, we define the learning rate

```
lr=4
```

and the initial match

```
mathc=0
```

We now need to put everything together. Use a text file to organize everything

First, set up the network (make sure you have all of these commands in your source file).

```
input=array(dim=4)
hidden=array(dim=3)
output=array(dim=4)
hnet=hidden
onet=output
wih=matrix(runif(12,-.3,.3),ncol=3,nrow=4)
oih=dih=matrix(rep(0,12),ncol=3,nrow=4)
who=matrix(runif(12,-.3,.3),ncol=4,nrow=3)
oho=dho=matrix(rep(0,12),ncol=4,nrow=3)
deltao=c(0, 0, 0, 0)
deltah=c(0, 0, 0)
stimuli=rbind(c(1,0,0,0),c(0,1,0,0),c(0,0,1,0),c(0,0,0,1),c(1,1,1,1), c(0,1,1,0))
lr=4
mathc=0
```

Now, we have to teach the network the first 3 patterns

```
#teach first 3 patterns
while(mathc<11.7){
  mathc=0
  for(k in 1:3){
  teach=stimuli[k,]
  input=teach
```

Here, include the code for the feed forward from input to hidden.

Then from hidden to output.

Find the delta o

Update the h-o weights

Find the deltah

Update the i-h weights

```
    # these lines print the output, calculate the match and close the loop
    print(round(output,2))
    mathc=mathc+sum(((2*output)-1)*((2*teach)-1))
  }
  print(mathc)
}

readline("First three patterns learned\n Press enter to learn next pattern")
#teach last pattern
mathc=0
while(mathc<3.85){
    mathc=0
    for(k in 4:4){
    teach=stimuli[k,]
    input=teach
```

Here, include the code for the feed forward from input to hidden.

Then from hidden to output.

Find the delta o

Update the h-o weights

Find the deltah

Update the i-h weights

```
    print(round(output,2))
    mathc=mathc+sum(((2*output)-1)*((2*teach)-1))
  }
  print(mathc)
}
readline("Final patterns learnt\n Press enter to begin test")


## test
for(k in 1:6){
    mathc=0
    teach=stimuli[k,]
    input=teach
#    print(input)
```

```
    print(c("input :",round(teach,2)))
```

This is the test, just feedforward the activation

```
  print(c("output:",round(output,2)))
  mathc=mathc+sum(((2*output)-1)*((2*teach)-1))

  print(c("Match (dot product) = ", mathc))
}
```