Applied Data Mining (UN3106)
Spring 2018
http://stat.columbia.edu/∼porbanz/UN3106S18.html

Peter Orbanz
porbanz@stat.columbia.edu
Phyllis Wan
pw2348@columbia.edu

Homework 5

Due: 26 April 2018

Homework submission: We will collect your homework **at the beginning of class** on the due date. If you cannot attend class that day, you can leave your solution in Phyllis Wan's postbox in the Department of Statistics, 10th floor SSW, at any time before then.

We do not accept homework submitted late. There will be no exceptions.

Problem 1

The purpose of this homework to understand how neural networks combine simple functions to represent more complicated ones. To do so, we look at a few simple networks, and use R to plot the functions they represent for different settings of the constituent functions (sigmoid functions and the "ramp function" or "rectified linear unit" $\phi(x) = \max\{0, x\}$).

We first consider the following network, for four different settings of the weights in the upper layer as listed in the table on the right:

$$\begin{tikzcd}
& x \arrow[dl, "w_1"'] \arrow[dr, "w_2"] & \\
\phi_1 \arrow[dr, "1"'] & & \phi_2 \arrow[dl, "1"] \\
& \sigma \arrow[d] & \\
& f(x) &
\end{tikzcd}$$

Plot	w_1	w_2
(i)	1	1
(ii)	1	$\frac{1}{2}$
(iii)	1	-1
(iv)	1	$-\frac{1}{2}$

In each of the following two cases, plot the function f represented by the network on the interval $[-10, 10]$, for each setting (i)–(iv) of the weights:

1. $\phi_1(x) = \phi_2(x) = \sigma(x)$.
2. $\phi_1(x) = \phi_2(x) = \max\{0, x\}$.

Hints:

- Here σ denotes the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.
- To plot a function $f(x)$, you can use the following codes.

```
# specify your function
f <- function(x){}
# generate a sequence of x values
x.vec <- seq(-10,10,length.out=1001)
# generate a vector of f(x) for each x in x.vec
fx.vec <- sapply(x.vec,f)
```

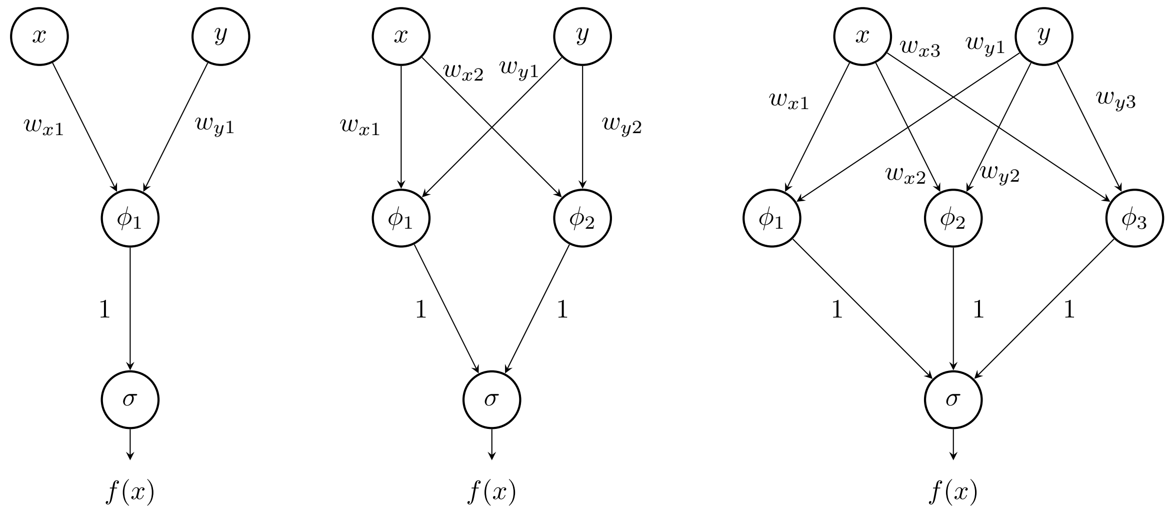
```
# plot each pair (x,f(x)), connect them into a line
plot(x.vec,fx.vec,type='l')
```

- For each case, produce a big plot with 4 subplots using the following codes.

```
par(mfrow=c(2,2))
plot(...)
plot(...)
plot(...)
plot(...)
```

Problem 2

Now we increase the dimension of the input space to two (that is, we use two input units), and try to visualize how the number of inner units affects the expressiveness of the network. We consider networks with one, two and three inner units. That requires quite a lot of weights, so instead of hand-tuning these weights, we randomize them and plot the resulting (random) functions.



- Generate the six weights $w_{x1}, w_{x2}, w_{x3}, w_{y1}, w_{y2}, w_{y3}$ randomly from a standard normal distribution.
- For these settings of the weights, plot the function represented by each of the three neural networks above above on $[-10, 10]^2$. Use the same list of weights for all networks, i.e. the first network uses only two of the six random numbers, and the second one only four.

1. Plot the three networks as described above, with $\phi_1 = \phi_2 = \phi_3 = \sigma$. Repeat for 4 sets of random weights.
2. Plot the three networks again, this time for $\phi_1(z) = \phi_2(z) = \phi_3(z) = \max\{0, z\}$. Repeat for 4 sets of random weights.

Hints:

- To plot a function $f(x, y)$ in 3D:

```
# specify your function
f <- function(x,y){}
# generate a sequence of x
x.vec <- seq(-10,10,length.out=51)
# generate a sequence of y
y.vec <- seq(-10,10,length.out=51)
# generate a matrix of f(x,y) with each comibination of (x,y) in x.vec and y.vec
fxy.mat <- matrix(NA,51,51)
for (i in 1:51){
  for (j in 1:51){
    fxy.mat[i,j] <- f(x.vec[i],y.vec[j])
  }
}
# plot each triple (x,y,f(x,y)) in 3D, connect them into a surface
persp(x.vec,y.vec,fxy.mat, theta = 45, phi = 15, col = "lightblue")
```

- For each set of weights, put your 3 plots on the same line:

```
par(mfrow=c(1,3))
plot(...)
plot(...)
plot(...)
```