1. (a) $Z^H = W^H x + b^H$

$$\Rightarrow Z^H = \begin{bmatrix} x_1+x_3 \\ x_2+x_3 \\ x_1+x_2-1 \\ x_1+x_2+x_3+1 \end{bmatrix}$$

The activation outputs in the ~~d~~ hidden layer are:

$$U^H = g_{act}(Z^H) = \begin{bmatrix} g_{act}(x_1+x_3) \\ g_{act}(x_2+x_3) \\ g_{act}(x_1+x_2-1) \\ g_{act}(x_1+x_2+x_3+1) \end{bmatrix} = \begin{bmatrix} 1\{x_1+x_3 \geq 0\} \\ 1\{x_2+x_3 \geq 0\} \\ 1\{x_1+x_2-1 \geq 0\} \\ 1\{x_1+x_2+x_3+1 \geq 0\} \end{bmatrix}$$

(b).

$$Z^o = W^o U^H + b^o = [1, 1, -1, -1] \begin{bmatrix} 1\{x_1+x_3 \geq 0\} \\ 1\{x_2+x_3 \geq 0\} \\ 1\{x_1+x_2-1 \geq 0\} \\ 1\{x_1+x_2+x_3+1 \geq 0\} \end{bmatrix} - 1.5$$

$$= 1\{x_1+x_3 \geq 0\} + 1\{x_2+x_3 \geq 0\} - 1\{x_1+x_2-1 \geq 0\} - 1\{x_1+x_2+x_3+1 \geq 0\} - 1.5$$

Thus in the region $x_1+x_3 \geq 0$, $x_2+x_3 \geq 0$, $x_1+x_2-1 < 0$, $x_1+x_2+x_3+1 < 0$

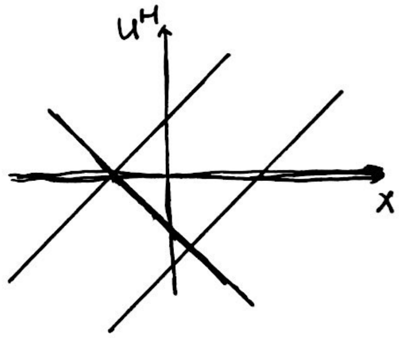
$Z^o = 1+1-0-0-1.5 = 0.5$.

$\hat{y} = 1$.

2. (a) $N_h = 3$.

$$z_j^H = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix}$$

$$u_j^H = g_{act}(z^H) = \begin{bmatrix} g_{act}(-x-1) \\ g_{act}(x+1) \\ g_{act}(x-2) \end{bmatrix}$$



(b) Given the ~~out~~ activation function $g_{out}(z^o) = z^o = \hat{y}$.

loss function factor: $g_{loss}(z_i^o, y_i) = \|z_i^o - y_i\|^2$

$\Rightarrow$ Loss function: $L(\theta) = \sum_{i=1}^{N} g_{loss}(z_i^o, y_i)$.

hw7_2cde

April 15, 2018

In [17]:
```
from sklearn import linear_model
import numpy as np
x=np.array([-2,-1,0,3,3.5])
y=np.array([0,0,1,3,3])
wH=np.array([-1,1,1])
bH=np.array([-1,1,-2])
x.shape=(1,5)
y.shape=(5,1)
wH.shape=(3,1)
bH.shape=(3,1)
zH=wH.dot(x)+bH
uH=[]
for i in range(0,len(zH)):
    temp=[]
    for j in range(0,len(zH[0])):
        temp.append(max(0,zH[i][j]))
    uH.append(temp)
uH=np.array(uH)
uH
```

Out[17]:
```
array([[ 1. ,  0. ,  0. ,  0. ,  0. ],
       [ 0. ,  0. ,  1. ,  4. ,  4.5],
       [ 0. ,  0. ,  0. ,  1. ,  1.5]])
```

In [18]:
```
a=np.ones(5)
a.shape=(5,1)
x=np.hstack((a,uH.T))
regr=linear_model.LinearRegression()
regr.fit(x,y)
coef=regr.coef_[0]
coef
```

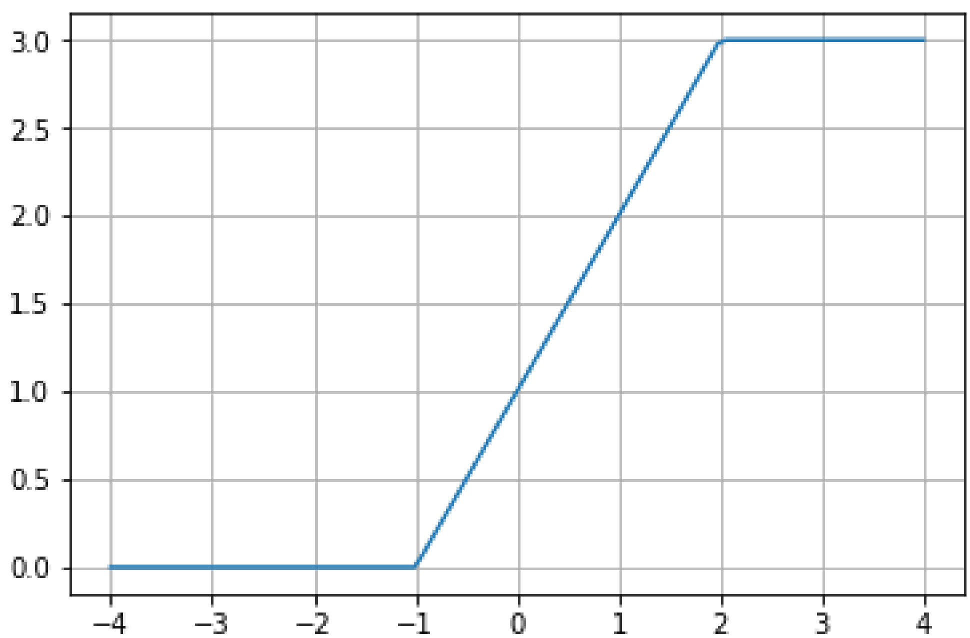
Out[18]:
```
array([  0.00000000e+00,   3.95662946e-16,   1.00000000e+00,
        -1.00000000e+00])
```

In [19]:
```
#b0=0,w0=[3.95662946e-16,1,-1]
```

In [22]:
```
import matplotlib.pyplot as plt
x1=np.linspace(-4,4,100)
```

```
x1.shape=(1,100)
zHts=wH.dot(x1)+bH
uHts=[]
for i in range(0,len(zHts)):
    temp=[]
    for j in range(0,len(zHts[0])):
        temp.append(max(0,zHts[i][j]))
    uHts.append(temp)
uHts=np.array(uHts)
b=np.ones(100)
b.shape=(100,1)
xts=np.hstack((b,uHts.T))
yhat=regr.predict(xts)
x2=x1.T
plt.plot(x2,yhat)
plt.grid()
plt.show
```

Out[22]: <function matplotlib.pyplot.show>



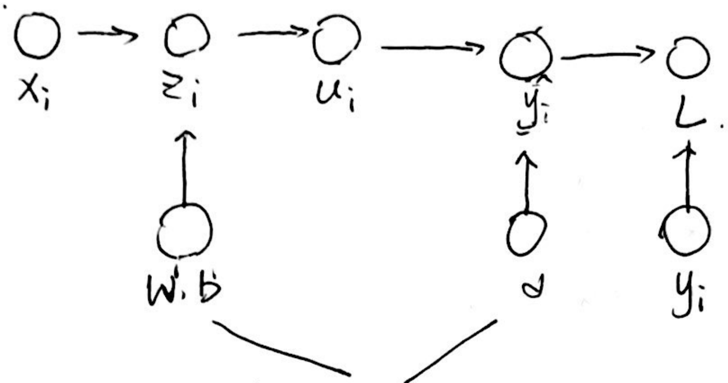
In [ ]:
```
def predict(x):
    wH=np.array([-1,1,1])
    bH=np.array([-1,1,-2])
    b0=0
    w0=[3.95662946*10**(-16),1,-1]
    wH.shape=(3,1)
```

```
bH.shape=(3,1)
w0.shape=(3,1)
zH=wH.dot(x.T)+bH
uH=max(zH,0)
a=np.ones(len(x))
a.shape=(len(x),1)
x=np.hstack(a,uH.T)
yhat=xts.dot(w0)+b0
return yhat
```

3. (a). $z_{ij} = \sum_{k=1}^{N_i} w_{jk} x_{ik} + b_j$, $u_{ij} = 1/(1+\exp(-z_{ij}))$, $i=1, \cdots N$, $j = 1, \cdots M$

$\hat{y} = \sum_{j=1}^{M} a_j u_{ij} \Big/ \sum_{j=1}^{M} u_{ij}$, $i=1 \ldots M$.

(b).



These two nodes are trainable nodes.

(c) $dL/d\hat{y}_i = \dfrac{d\left(\sum_{i=1}(y_i - \hat{y}_i)^2\right)}{d\hat{y}_i} = 2(\hat{y}_i - y_i)$.

(d) $dL/du = dL/d\hat{y} \cdot d\hat{y}/du$ according to the chain rule.

(e) $dL/dz = dL/du \cdot du/dz$.

(f) $dL/dw_{jk} = dL/dz \cdot dz/dw_{jk}$.

$dL/db_j = dL/dz \cdot dz/db_j$

(g) $dL/dw_{jk} = dL/d\hat{y}_i \cdot d\hat{y}_i/du_{ij} \cdot du_{ij}/dz_{ij} \cdot dz_{ij}/dw_{jk}$

$dL/db_j = dL/d\hat{y}_i \cdot d\hat{y}_i/du_{ij} \cdot du_{ij}/dz_{ij} \cdot dz_{ij}/db_j$.

hw7_3h

April 16, 2018

In [ ]:
```
u1=np.sum(u,axis=1)
u2=np.sum(u*alpha[None,:],axis=1)
dy_du=(u1-u2)/(u1**2)
dL_du=dL_dy[:,None]*dy_du
```