Linear mapping
Neural mapping

Numerical and Symbolic approaches

array

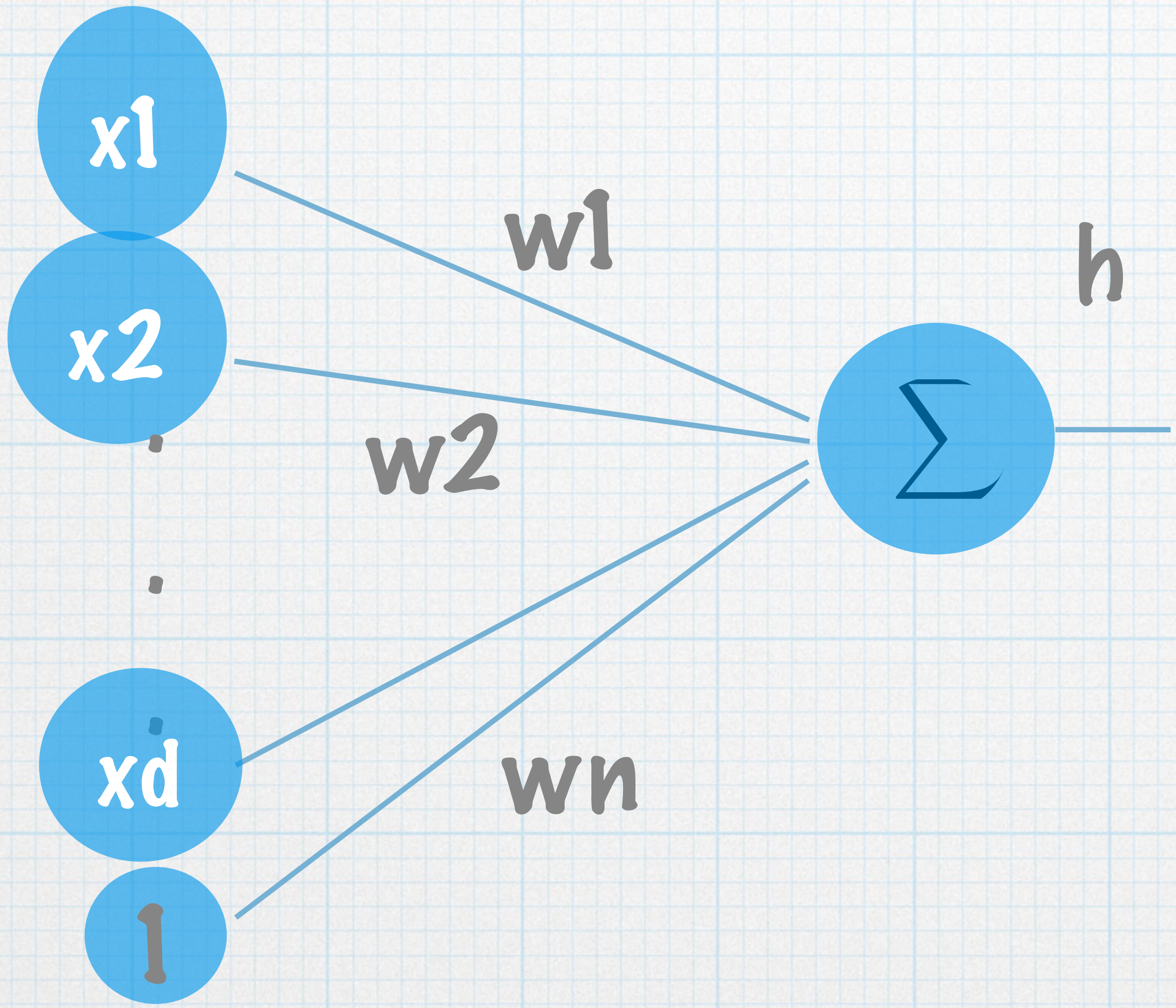
A = sym('a',[1 3])

A =

[a1, a2, a3]

```
>> A = sym('a',[2 3])

A =

[ a1_1, a1_2, a1_3]
[ a2_1, a2_2, a2_3]

>> A(1,:)

ans =

[ a1_1, a1_2, a1_3]
```

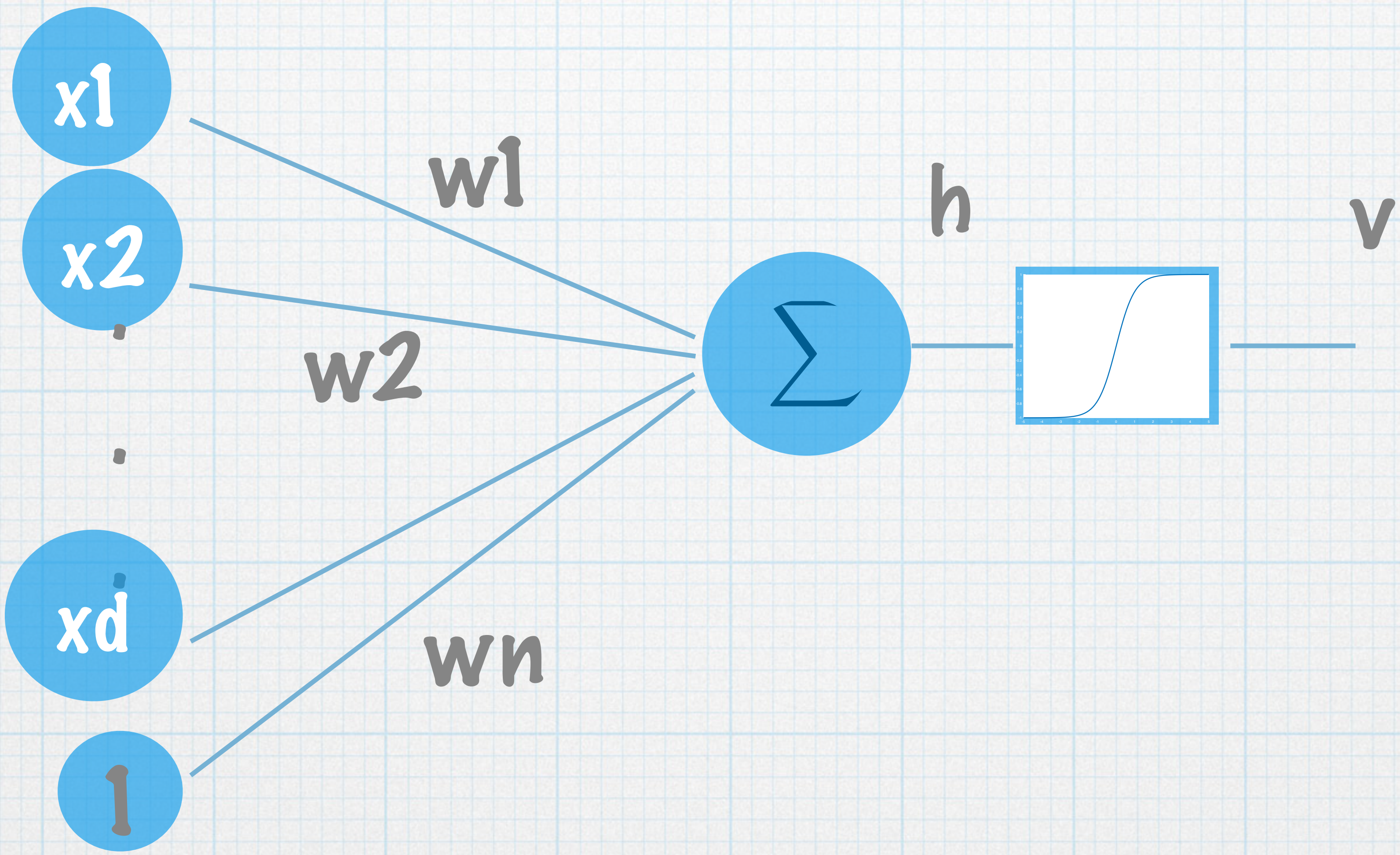
Linear projection



```
x=sym('x',[3 1]);
w=sym('w',[3 1]);
s=char(sum(w.*x));
ss=vector_input(s,'w','x');
f=inline(ss,'x','w')
f[1 1 1],[2 2 2])
```

```
function ss=vector_input(s,c1,c2)
    L=length(s);i=1;ss=[];
    while i <= L-1
        if ((s(i)==c1 | s(i)==c2) & (s(i+1)<='9' & s(i+1) >= '0'))
            start_i=i;
            while i<=L-1 & s(i+1) <= '9' & s(i+1) >= '0'
                i=i+1;
            end
            end_i=i;
            ss=[ss s(start_i) '(' s(start_i+1:end_i) ')'];
            i=i+1;
        else
            ss=[ss s(i)];
            i=i+1;
        end
    end
```

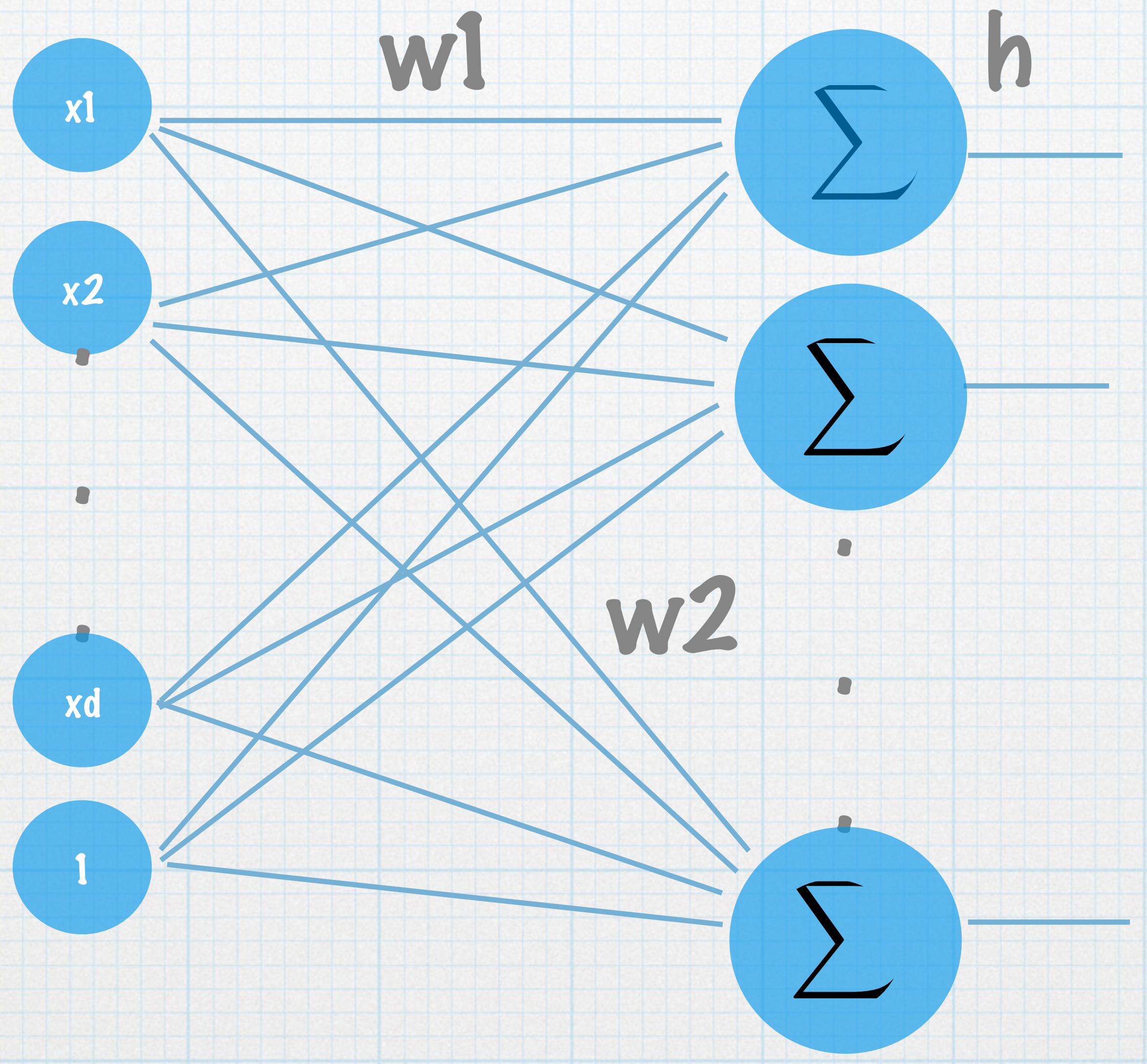
A perceptron



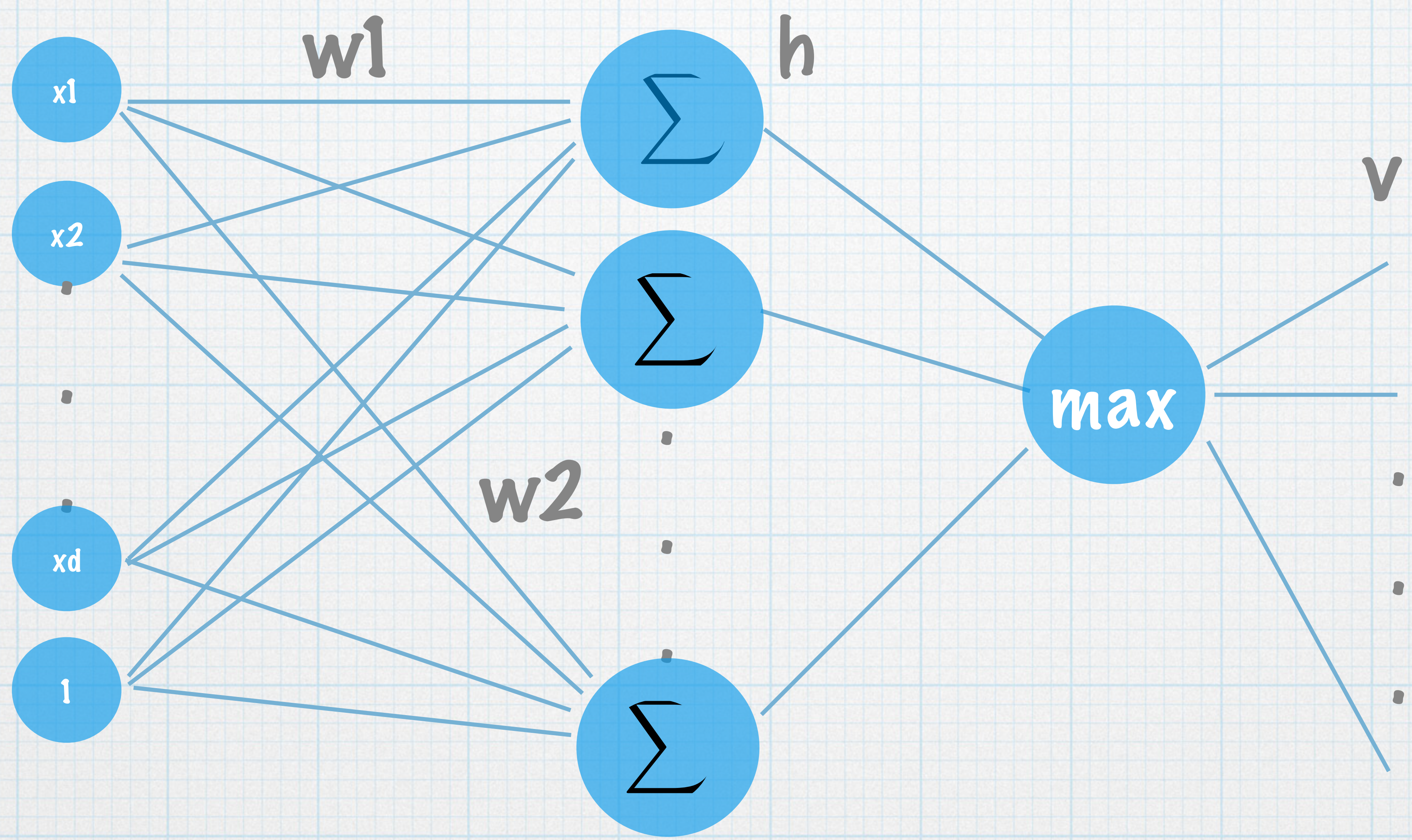
```
>> x=sym('x',[3 1])
s=char(sum(w.*x))
ss=vector_input(s,'w','x')

f=inline(ss,'x','w')


x =

 x1
 x2
 x3



s =

    'w1*x1 + w2*x2 + w3*x3'



ss =

    'w(1)*x(1) + w(2)*x(2) + w(3)*x(3)'



f =

     Inline function:
     f(x,w) = w(1)*x(1) + w(2)*x(2) + w(3)*x(3)

>> f([1 1 1],[2 2 2])

ans =

     6

>>
```

```
x=sym('x',[3 1]);
w=sym('w',[3 1]);
s=char(tanh(sum(w.*x)) );
ss=vector_input(s,'w','x');
f=inline(ss,'x','w')
f([1 1 1],[2 2 2])
```

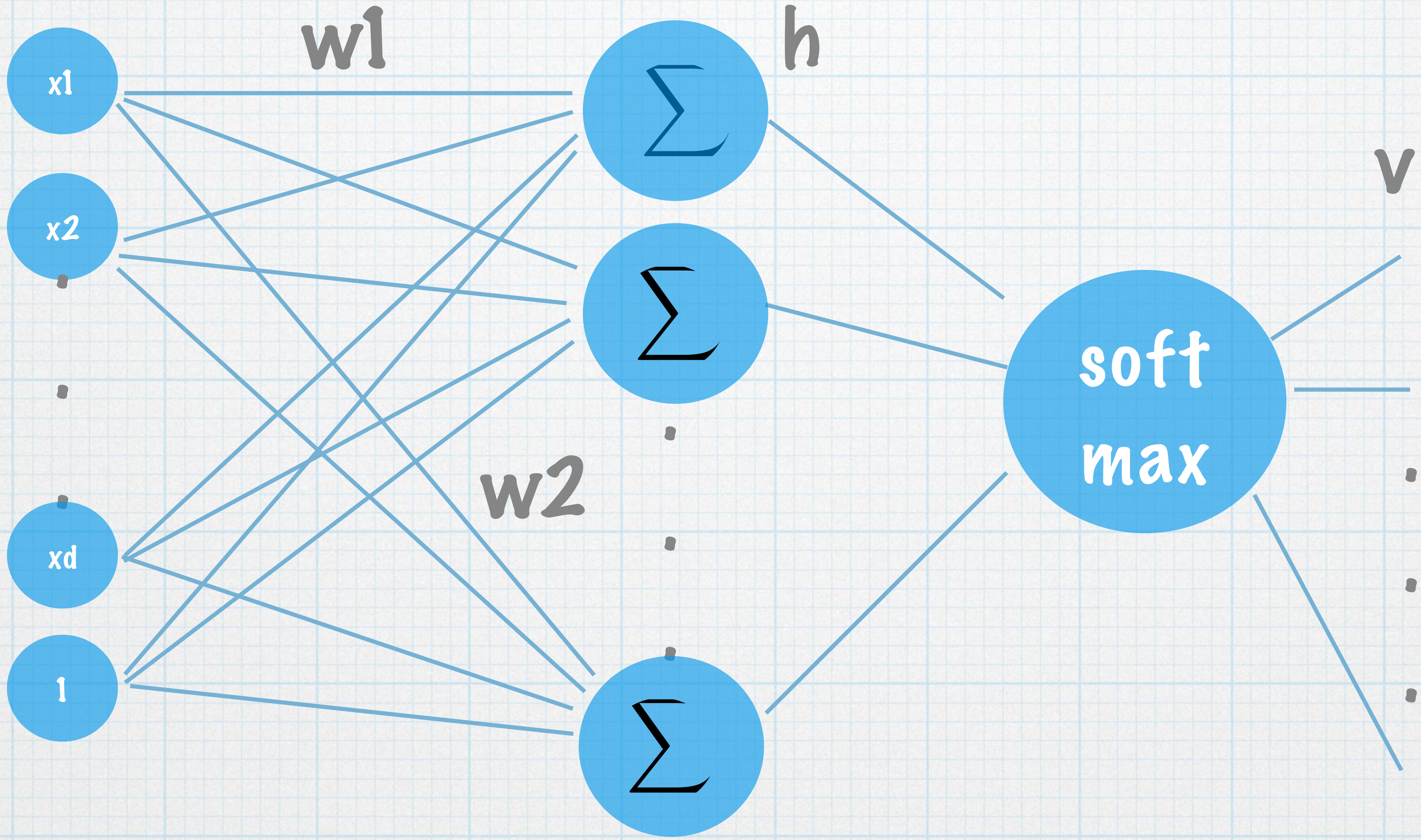
Linear mapping



Winner-take-all



Softmax

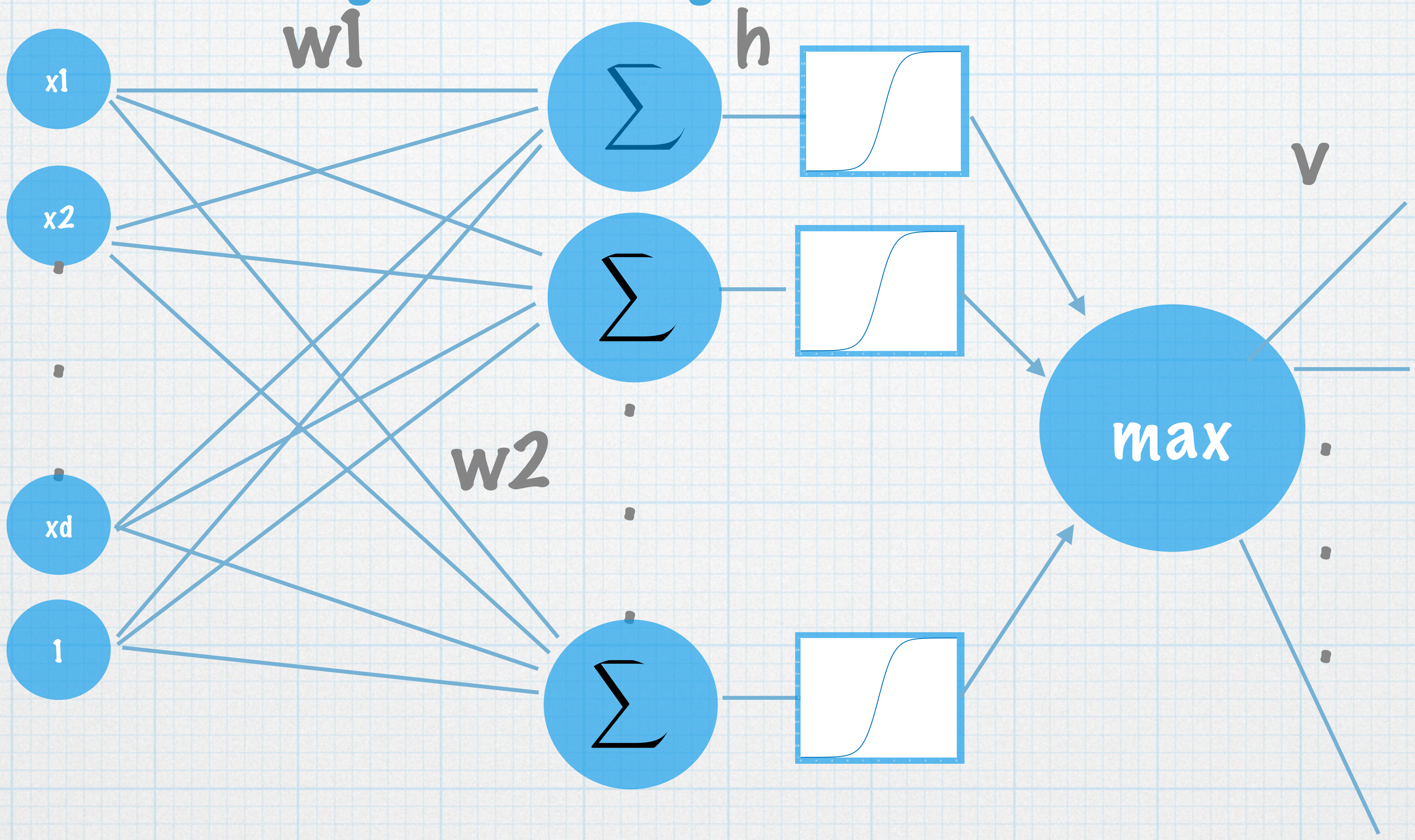


$h = [h_1, h_2, ...h_n]$

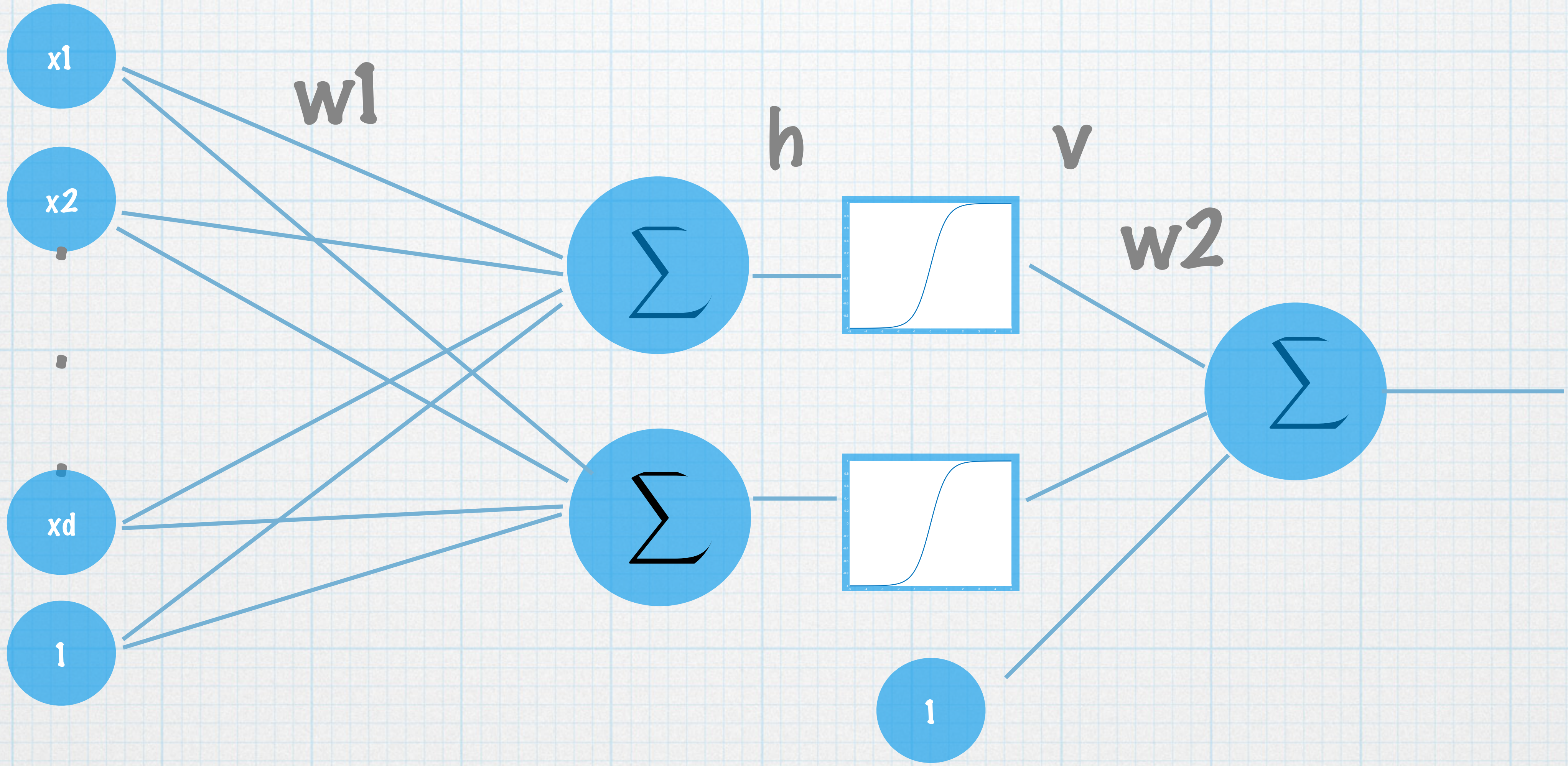
$v = [v_1, v_2, ..., v_n]$

$$v_i = \frac{exp(h_i)}{\sum_j exp(h_j)}$$

perceptrons



perceptrons

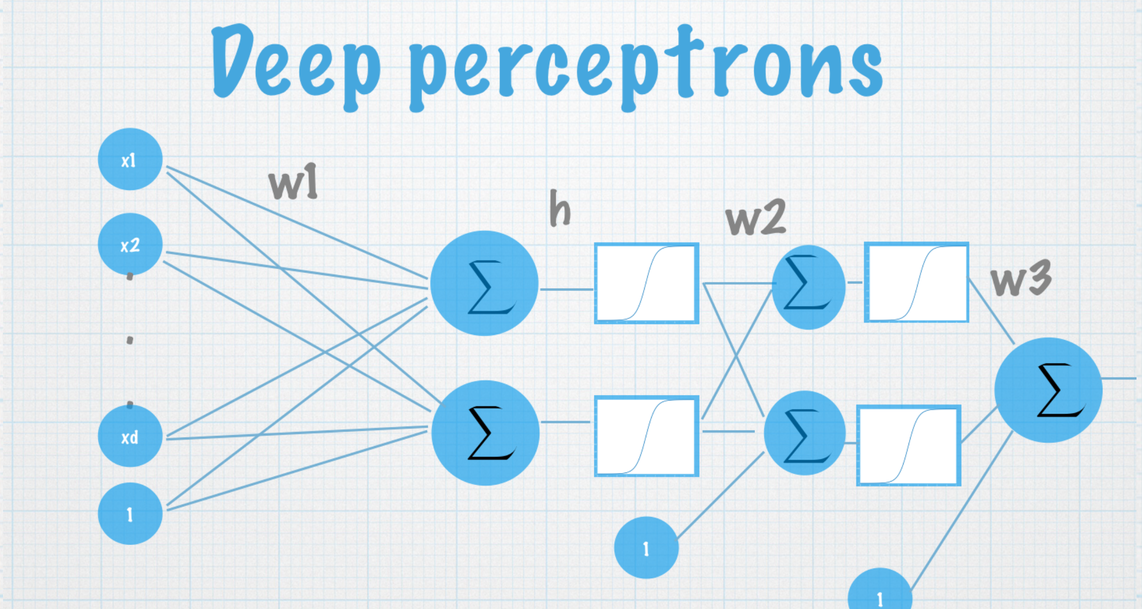


$$y = f(x) = w_2 \tanh(w_1 x)$$

object oriented programming

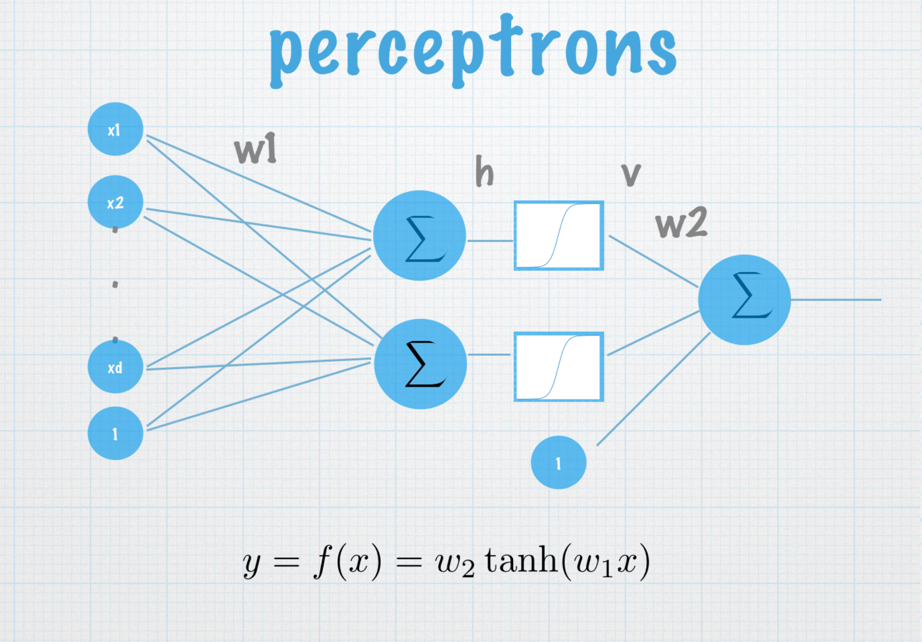
```
classdef perceptrons
    %UNTITLED12 Summary of this class goes here
    %   Detailed explanation goes here

    properties
    end

    methods
    end

end
```

```
classdef perceptrons
    %UNTITLED Summary of this class goes here
    %   Detailed explanation goes here

    properties
        layers;
        w;
        nL;
        a;
    end

    methods
    end
```

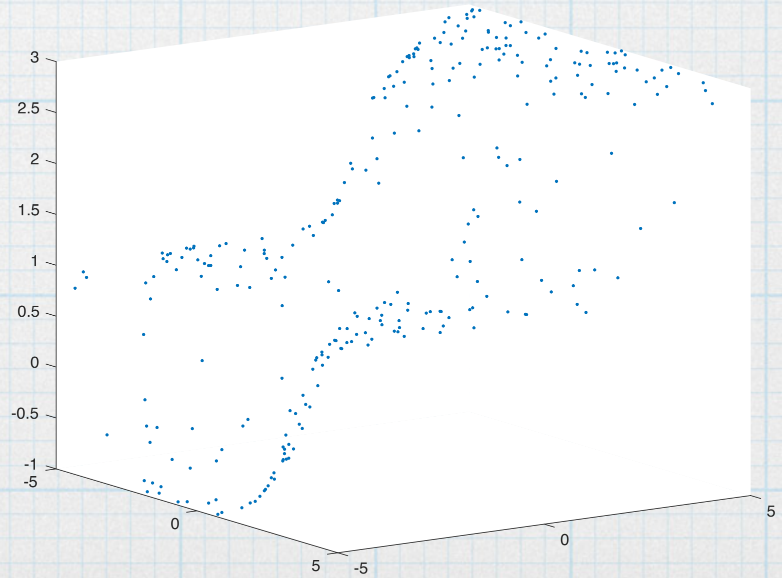


$$y = f(x) = w_3 \tanh(w_2 \tanh(w_1 x))$$

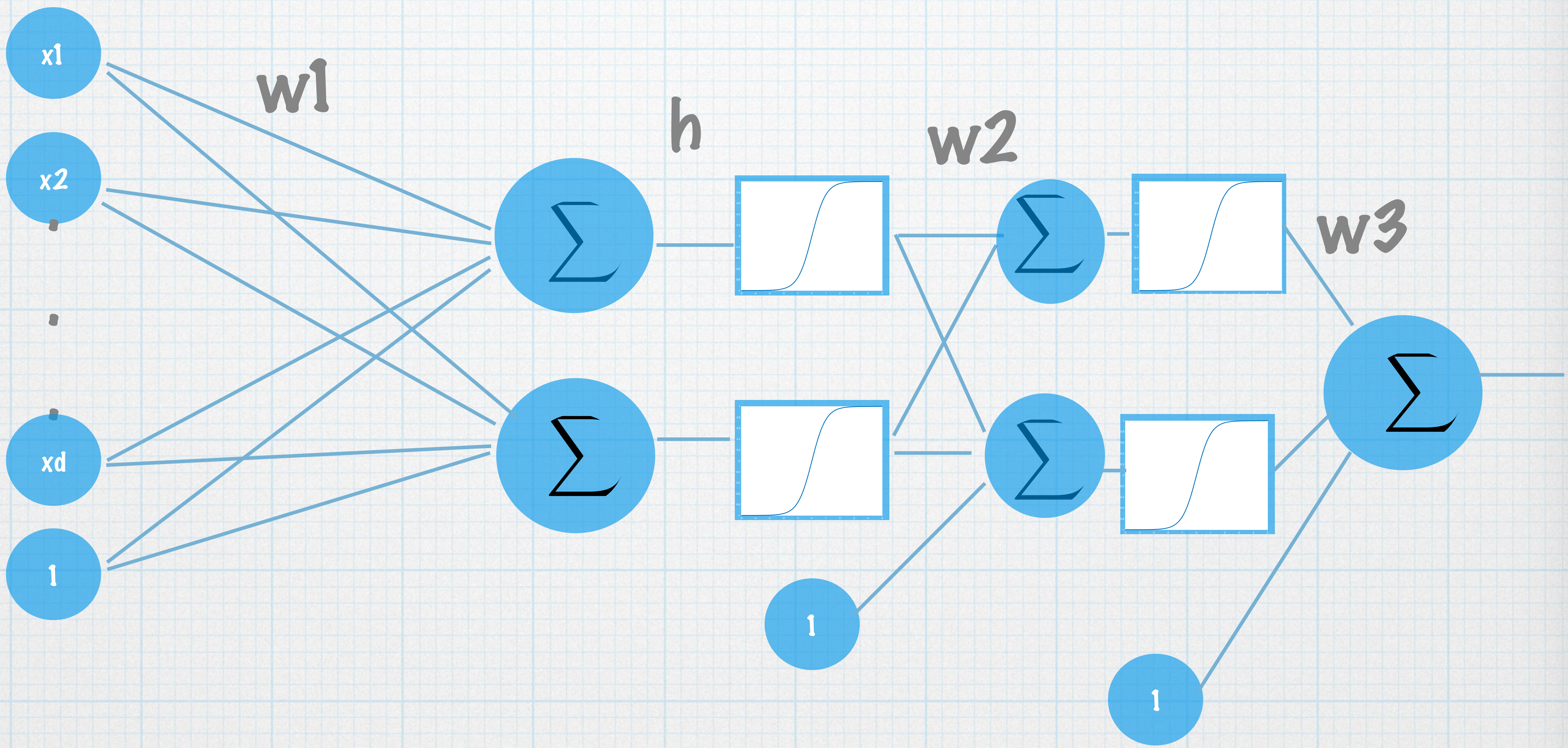
```
methods
    function obj=perceptrons(layers,w,nL)
        % layers include input, hidden and output layers
        obj.w=w;
        obj.nL=nL;
        obj.layers=layers;
    end
    function obj=ff(obj,x)
        % exercise today
        % .
        % .
        % .
    end
end
```




```
w{1}=[1 1 2;1 -1 -1]';nL{1}='tanh';
w{2}=[1 -1 1]';nL{2}='linear';
layers=3;

net=perceptrons(layers,w,nL);

x=rand(300,2)*10-5;
net=net.ff(x);
y=net.a{layers};
plot3(x(:,1),x(:,2),y,'.')
```

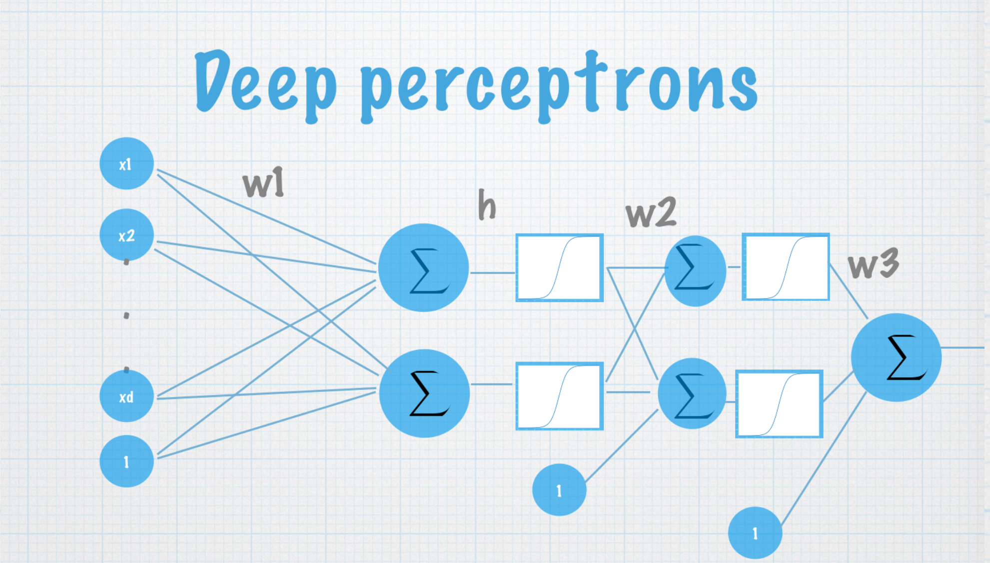


Deep perceptrons



$$y = f(x) = w_3 \tanh(w_2 \tanh(w_1 x))$$

```
w{1}=[1 1 2;1 -1 -1]';nL{1}='tanh';
w{2}=[1 -1 1;-1 -1 -1]';nL{2}='tanh';
w{3}=[1 1 1/2]';nL{3}='linear';
layers=4;

net=perceptrons(layers,w,nL);
x=rand(300,2)*10-5;
net=net.ff(x);
y=net.a{layers};
plot3(x(:,1),x(:,2),y,'.')
```



$$y = f(x) = w_3 \tanh(w_2 \tanh(w_1 x))$$