Jaringan Saraf Tiruan (REVIEW)

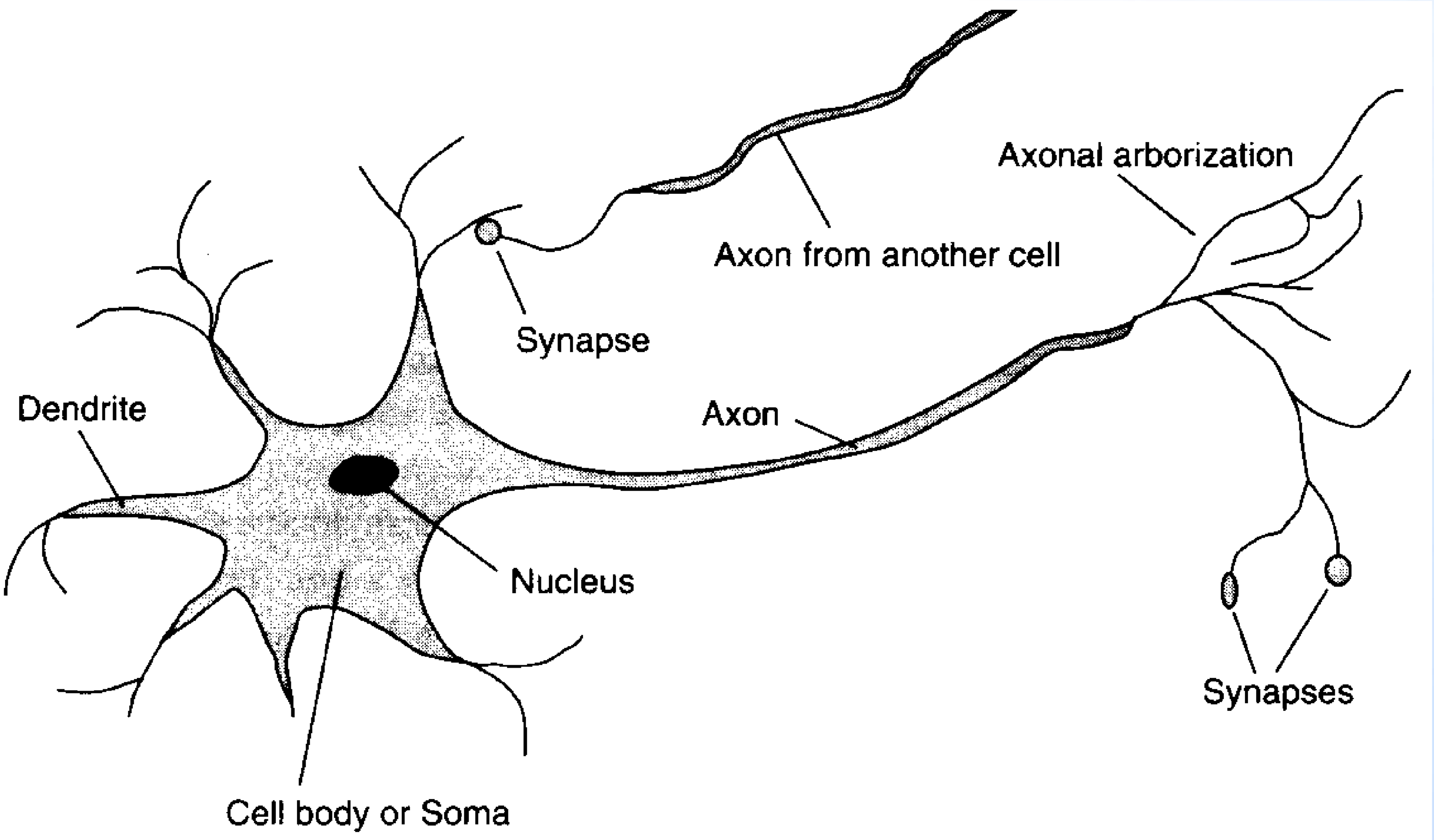
Review

- Review dasar teori NN
- Arsitektur JST
- Proses propagasi JST
- Training JST
- JST pada MATLAB

(Artificial) Neural Network

- Computational model inspired from neurological model of brain
- Human brain computes in different way from digital computer
  - highly complex, nonlinear, and parallel computing
  - many times faster than d-computer in
    - pattern recognition, perception, motor control
  - has great structure and ability to build up its own rules by experience
    - dramatic development within 2 years after birth
    - continues to develop afterward (Language Learning Device before 13 years old)
  - Plasticity : ability to adapt to its environment

Biological Neuron

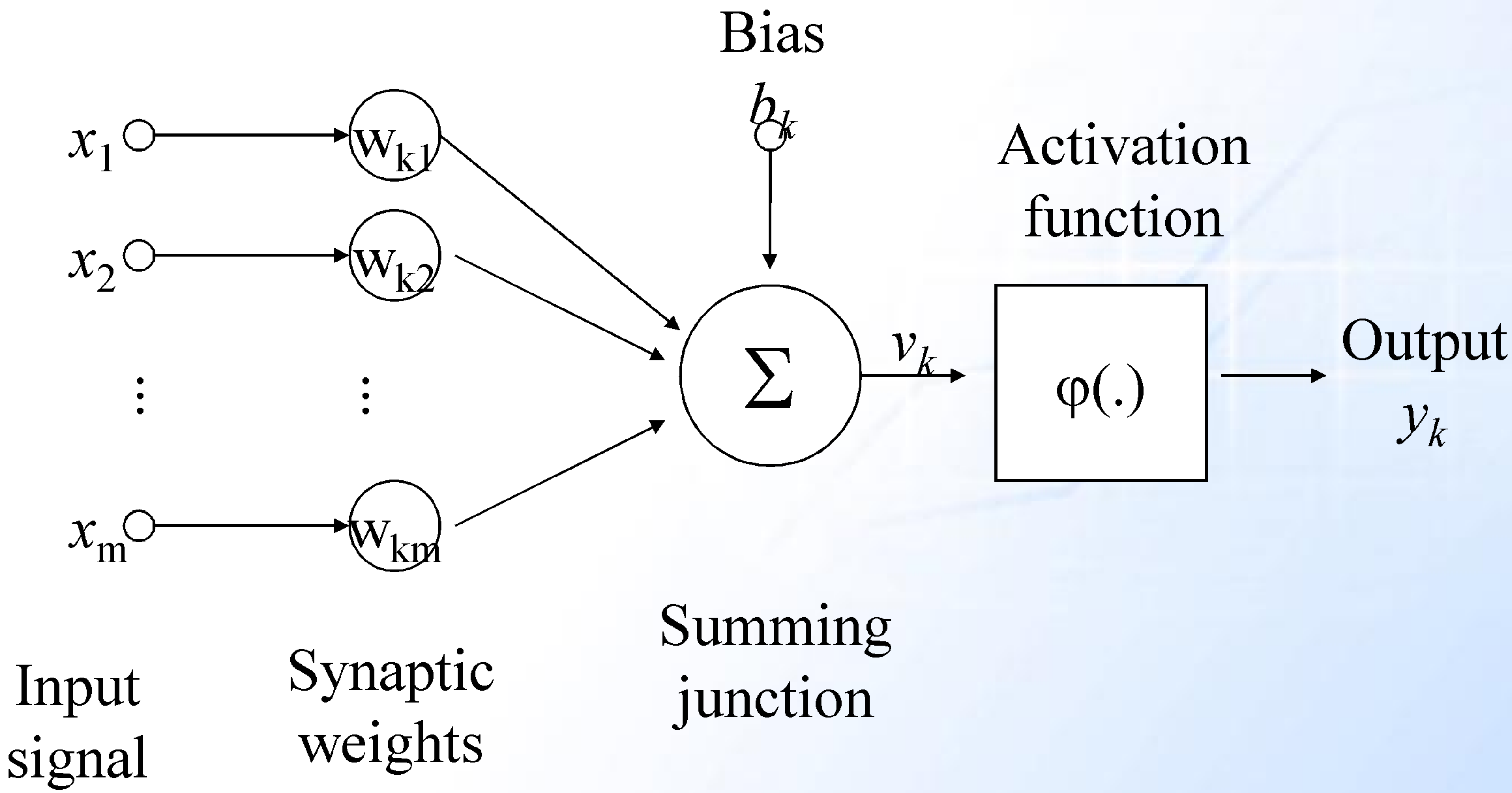


Excerpted from *Artificial Intelligence: Modern Approach*
by S. Russel and P. Norvig

Models of Neuron

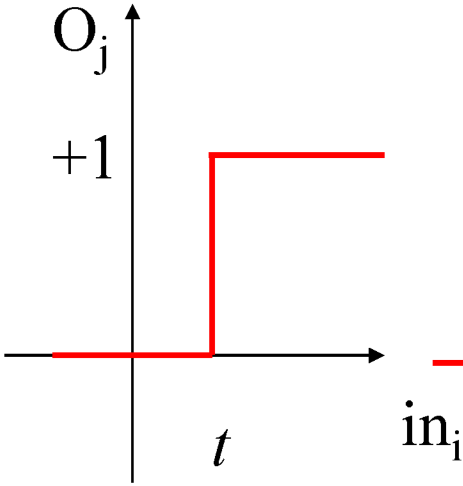
- Neuron is information processing unit
- A set of synapses or connecting links
  - characterized by weight or strength
- An adder
  - summing the input signals weighted by synapses
  - a linear combiner
- An activation function
  - also called squashing function
    - squash (limits) the output to some finite values

Nonlinear model of a neuron

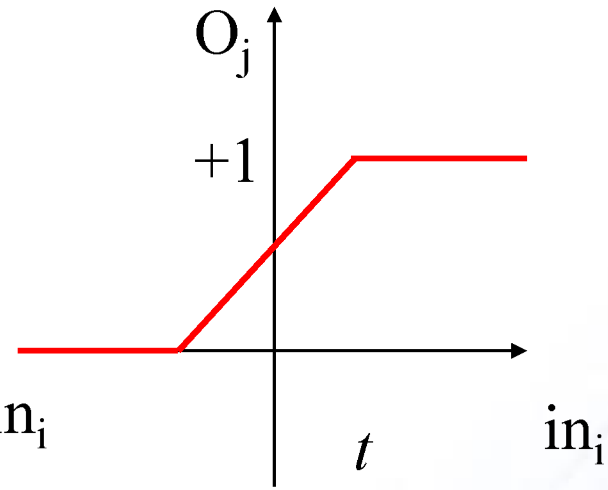


$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \qquad y_k = \varphi(v_k)$$

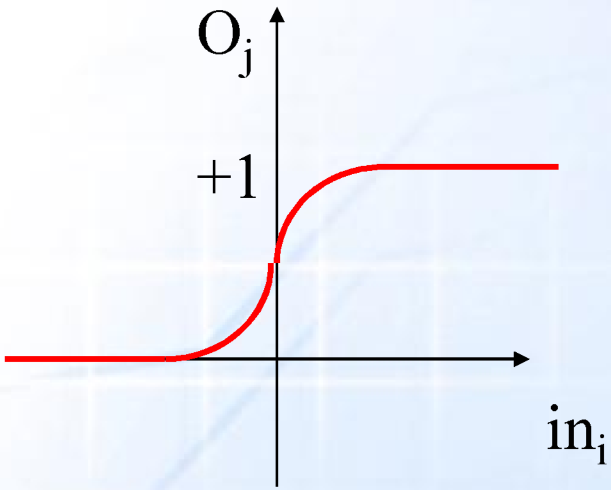
Types of Activation Function



Threshold Function



Piecewise-linear Function

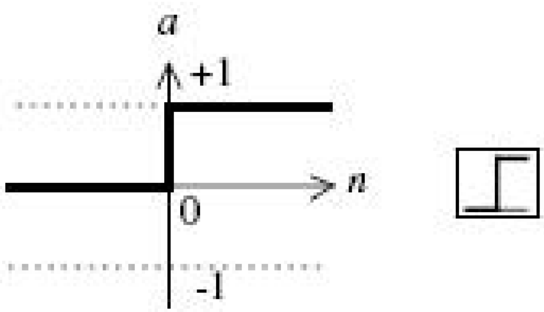


Sigmoid Function (differentiable)

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

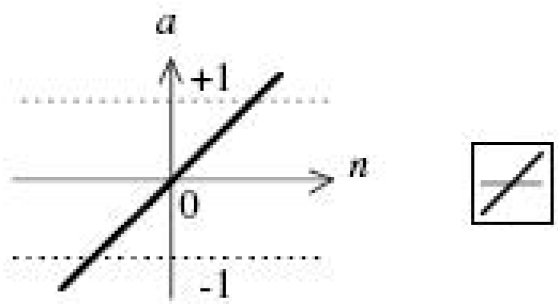
a is slope parameter

Another representation



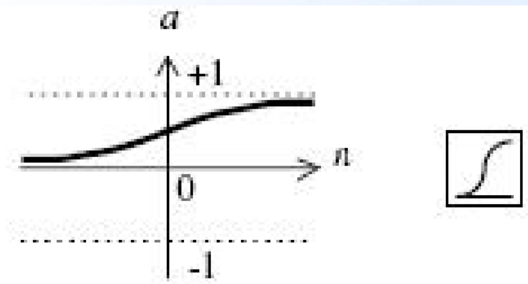
$a = hardlim(n)$

Hard-Limit Transfer Function



$a = purelin(n)$

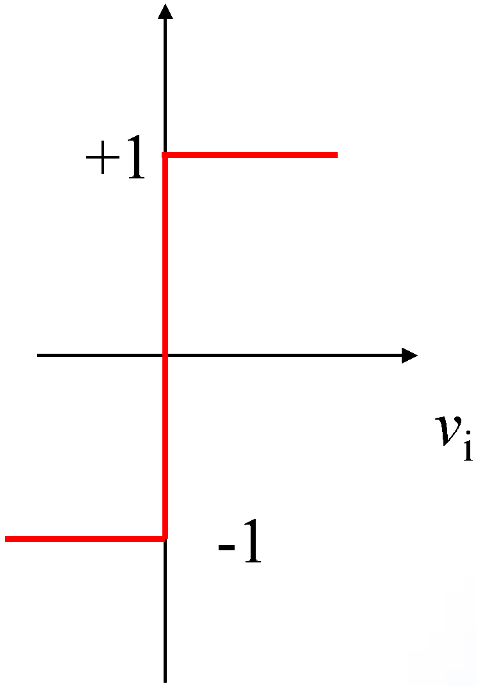
Linear Transfer Function



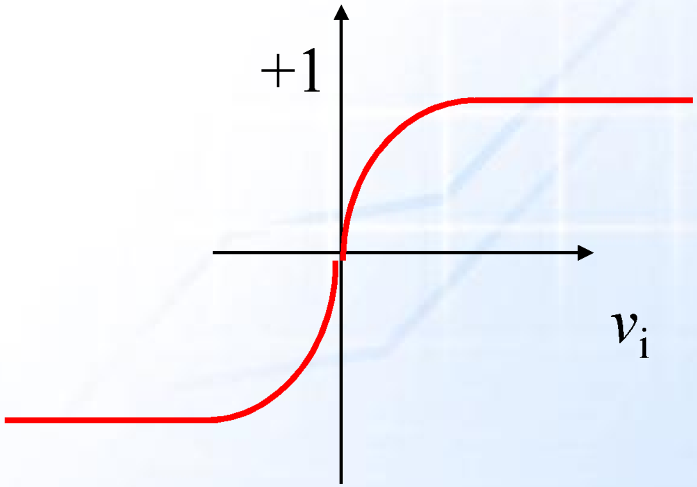
$a = logsig(n)$

Log-Sigmoid Transfer Function

Activation Function value range



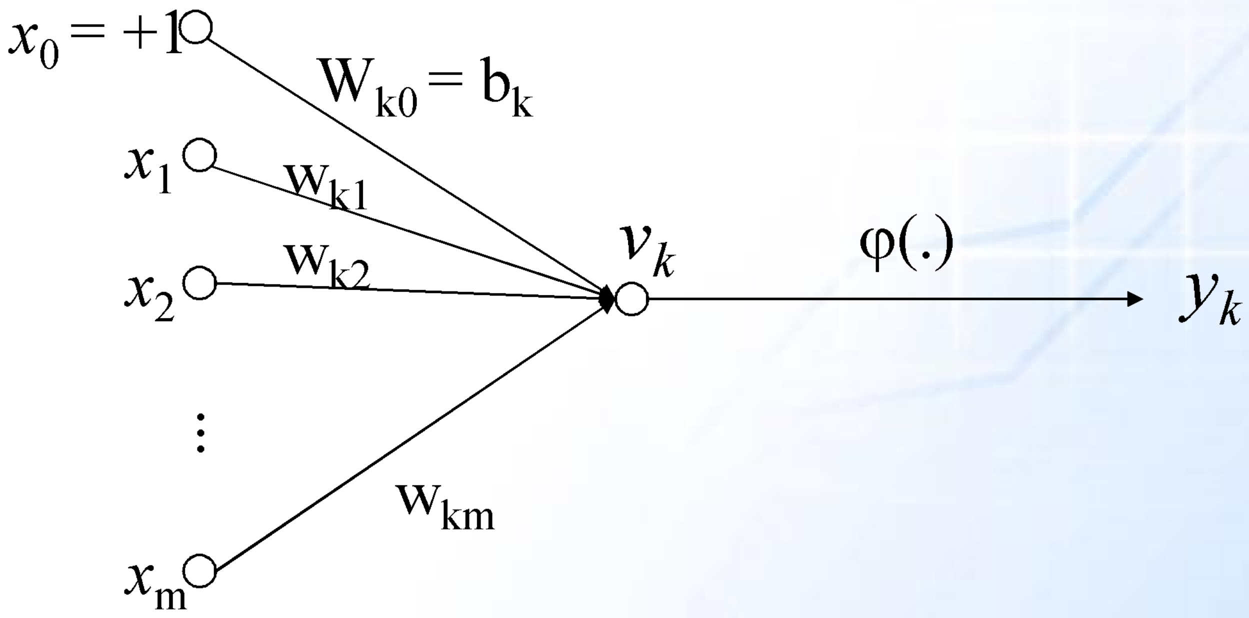
Signum Function



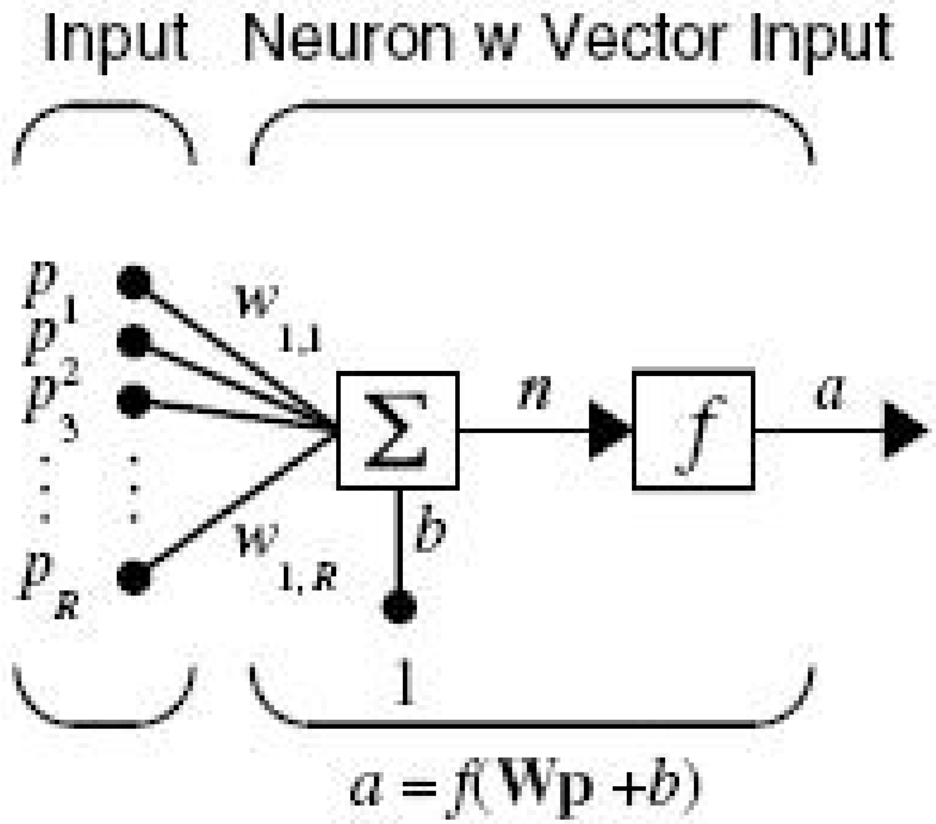
Hyperbolic tangent Function

$$\varphi(v) = \tanh(v)$$

Signal Flow Graph of a Neuron



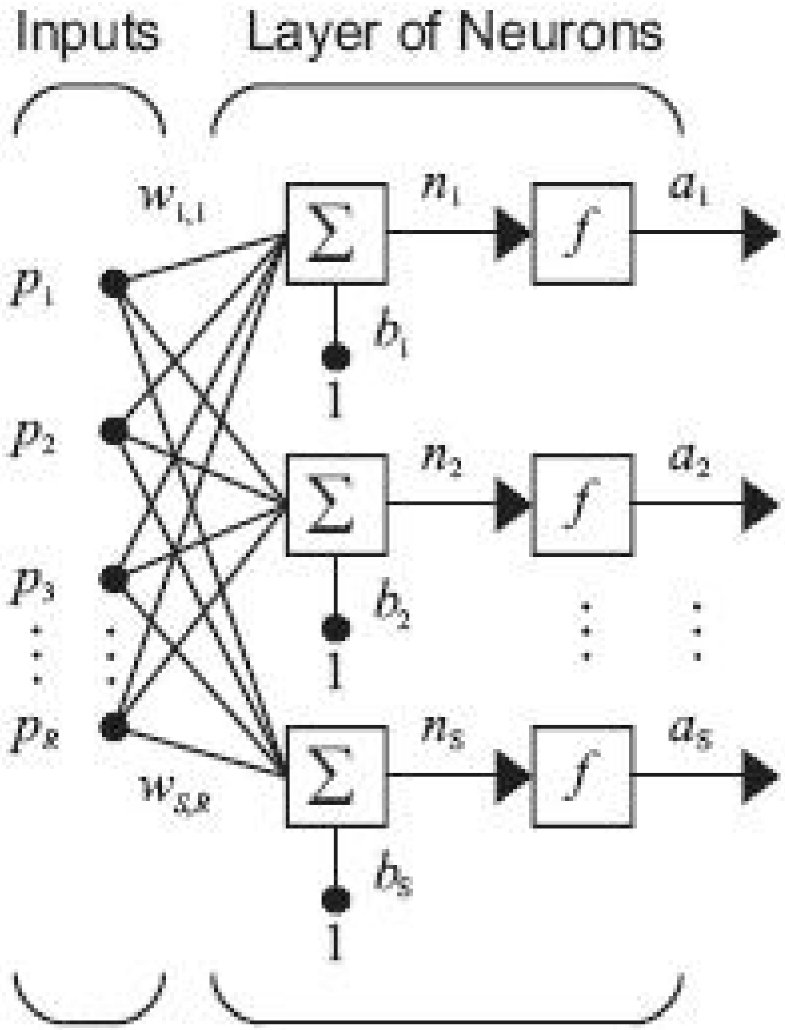
Neuron with vector input



Where

R = number of elements in input vector

A Layer Neuron



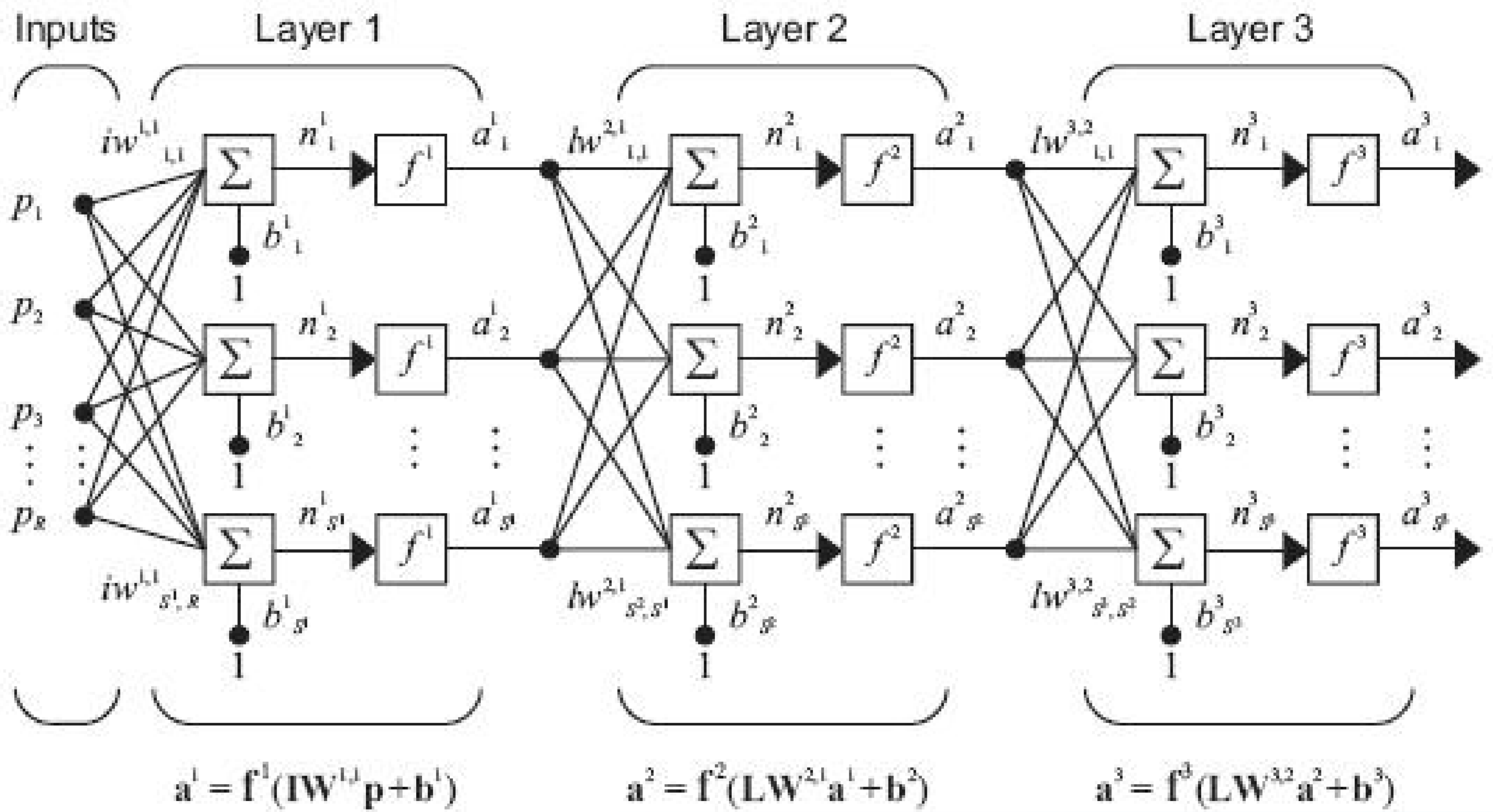
$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

Where

R = number of elements in input vector

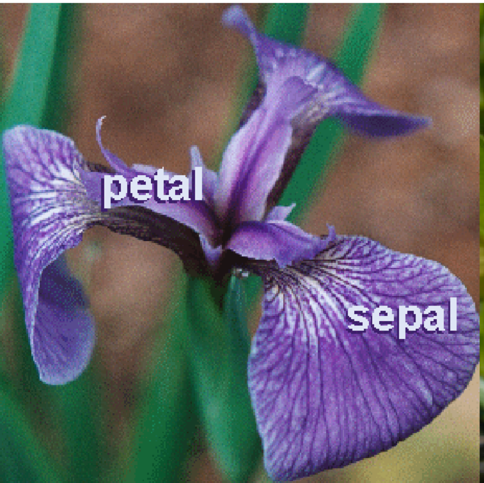
S = number of neurons in layer

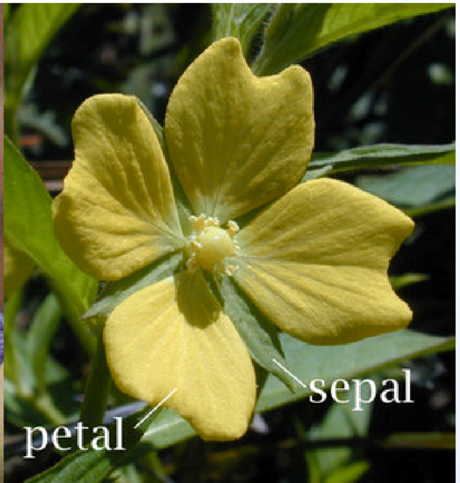
Multiple Layer Neuron



Example : Fisher's Iris Data

Type	PW	PL	SW	SL
0	2	14	33	50
1	24	56	31	67
1	23	51	31	69
0	2	10	36	46
1	20	52	30	65
1	19	51	27	58
2	13	45	28	57
2	16	47	33	63
1	17	45	25	49
2	14	47	32	70
0	2	16	31	48





- The table above gives Ronald Fisher's measurements of *type*, *petal width* (PW), *petal length* (PL), *sepal width* (SW), and *sepal length* (SL) for a sample of 150 irises. The lengths are measured in millimeters. Type 0 is*Setosa*; type 1 is *Verginica*; and type 2 is *Versicolor*.

NN Clasification in MATLAB

```
>> load fisheriris meas species
P=meas;
P=P';
T1=strcmpi(species,'setosa');
T2=2*strcmpi(species,'versicolor');
T3=3*strcmpi(species,'virginica');
T=T1+T2+T3;
T=T';
net=newff(P,T,5);
net=train(net,P,T);
Y=sim(net,P);
error=sum(abs(T-round(Y)))
```

```
>> xmaxi=net.inputs{1}.processSettings{3}.xmax
   7.9000
   4.4000
   6.9000
   2.5000
>> xmini=net.inputs{1}.processSettings{3}.xmin
   4.3000
   2.0000
   1.0000
   0.1000
```

```
>> ymaxi=net.inputs{1}.processSettings{3}.ymax
ymaxi = 1
```

- ```
  >> ymini=net.inputs{1}.processSettings{3}.ymin
  ymini =  -1
  ```
  ```

```
>> W1=net.IW{1}
W1 =
    4.3712   -3.6262   -1.9071   -1.1133
   -0.1448    0.2100   -1.0577    0.1524
   -0.8368    2.4485    1.3715   -3.2401
    6.4131   -7.4378   -0.1629    0.7238
   -0.2555    1.7087    0.1774   -1.5107
```

```
b1=net.b{1}
b1 =
  -2.1267
   0.6695
   0.8906
   1.3372
  -7.1393
```

```
>> W2=net.LW{2}
W2 =
  -0.5032   -1.2393   -0.4487    0.1026   -0.0399
>> b2=net.b{2}
b2 =
   0.1638
```

```
>> xmaxo=net.outputs{2}.processSettings{2}.xmax
xmaxo =     3
>> xmino=net.outputs{2}.processSettings{2}.xmin
xmino =     1
>> ymaxo=net.outputs{2}.processSettings{2}.ymax
ymaxo =     1
>> ymino=net.outputs{2}.processSettings{2}.ymin
ymino =    -1
```

Example

```
>> input = P(:,33)
ans =
    5.2000
    4.1000
    1.5000
    0.1000
>> sim(net,input)
ans =
    1.0085
>> T(33)
ans =
     1
```

sim(net,P(:,33)) ???

```
>> pNN = (ymaxi-ymini)*(input-xmini)./(xmaxi-xmini) + ymini;
>> n=W1*pNN+b1;
>> a=2./(1+exp(-2*n))-1;
>> yNN=W2*a+b2;
>> yfinal = (xmaxo-xmino) * (yNN-ymino)/(ymaxo-ymino) +
   xmino

yfinal =

   1.0085
```