

EE 434
02/19/07

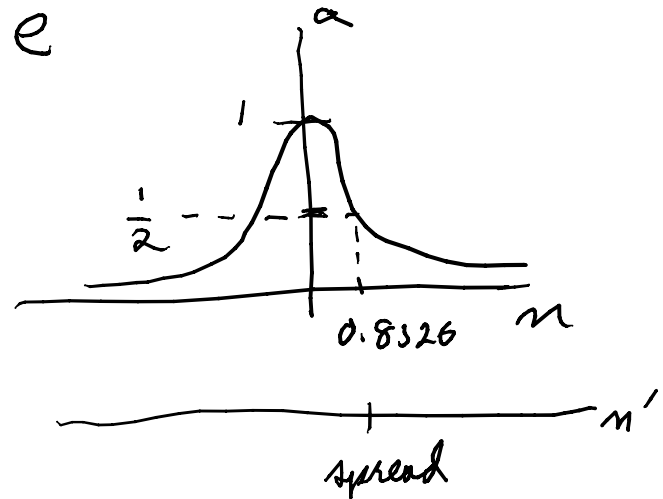
radial basis function - n^2
 $a = \text{radbas}(n) = e^{-n^2}$

$$a = \frac{1}{2} = e^{-n_{1/2}^2}$$

$$\ln\left(\frac{1}{2}\right) = -n_{1/2}^2$$

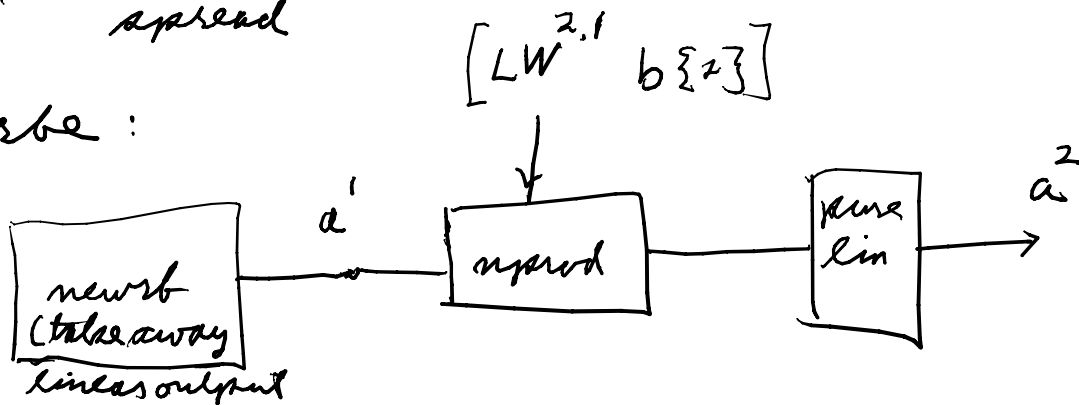
$$n_{1/2} = \sqrt{-\ln(1/2)} = 0.8326$$

$$n = \frac{0.8326}{\text{spread}} n'$$



$$\frac{n}{0.8326} = \frac{n'}{\text{spread}}$$

newrb:



$$[LW\{2,1\} \quad b\{2\}] * \begin{bmatrix} a\{1\} \\ \vdots \\ 1 \end{bmatrix} = T$$

solve this for $LW\{2,1\}$, $b\{2\}$

newrb solves for these via some type of pseudo-inverse

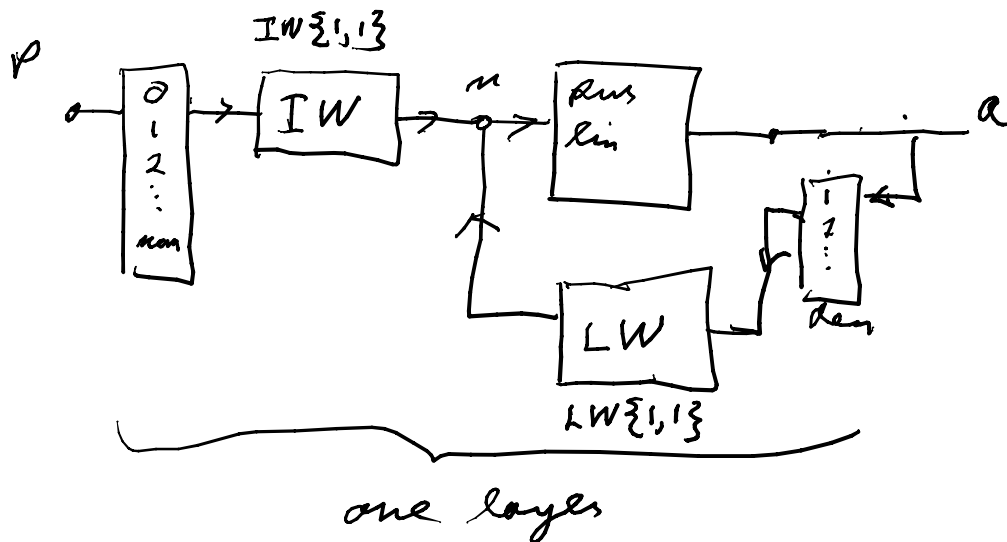
Digital filter; single input single output

$$a(z) = T(z) \cdot P \quad ; \quad T(z) = \frac{\sum_{i=0}^{num} a_i z^{-i}}{1 + \sum_{j=1}^{den} b_j z^{-j}}$$

$\frac{1}{z}$ = unit delay

$$\left(1 + \sum_{j=1}^{den} b_j z^{-j}\right) a = \left(\sum_{i=0}^{num} a_i z^{-i}\right) P$$

$$a = \left(\sum_{i=0}^{num} a_i z^{-i}\right) P - \left(\sum_{j=1}^{den} b_j z^{-j}\right) a$$



```
>> net1=network;
>> net1.numInputs=1;
>> net1.numLayers=1;
>> net1.inputConnect(1,1)=1;
>> net1.layerConnect(1,1)=1;
>> net1.outputConnect=[1];
>> net1.targetConnect=[1];
>> net1.inputs{1}.range=[-1 1];
>> net1.layers{1}.transferFcn='purelin';
>> net1.layers{1}.initFcn='initnw';
```

```
>> net1.initFcn='initlay';  
>> net1.performFcn='mse';  
>> net1.trainFcn='trainlm';  
>> net1.inputWeights{1,1}.delays=[0 1 2 3];  
>> net1.layerWeights{1,1}.delays=[ 1 2 3];  
>> net1.initFcn='initlay';  
>> net1.performFcn='mse';  
>> net1.trainFcn='trainlm';  
>> net1.trainParam.goal=1e-10;
```

```
net1=init(net1);  
>> P=[-0.5:0.1:0.5];  
>> T=[-0.2:0.1:0.8];  
>> P1=con2seq(P);  
>> T1=con2seq(T);  
>> net2=train(net1,P1,T1);  
>> Y=sim(net2,P1);  
>> Y2=seq2con(Y);  
>> plot(P,T,'-',P,Y2{1},' +r')
```