HUT , Institute of mathematics Gripenberg
Mat-1.196 Mathematics of neural networks
Exercise 6
19.2–27.2.2002

1. Let $\sigma(t) = 1$ if $t \geq 0$ and $\sigma = 0$ if $t < 0$. Construct a neural network with two inputs, one hidden layer with nodefunction σ such that with inputs $(0,1)$ and $(1,0)$ the output is 0 and with inputs $(1,1)$ and $(0,0)$ the output is 1.

Solution: Choose the matrix W_1 so that $W_1(i,j) = 1$ for all i and j. Choose $\tau_1(1) = \frac{3}{2}$ and $\tau_1(2) = \frac{1}{2}$ The output from the hidden layer is then $(\sigma(2-\tau_1(1)), \sigma(2-\tau_1(2)) = (1,1)$, $(\sigma(1-\tau_1(1)), \sigma(1-\tau_1(2)) = (0,1)$, or $(\sigma(-\tau_1(1)), \sigma(-\tau_1(2)) = (0,0)$.

Now we choose $W_2(1,1) = 1$, $W_2(1,2) = -1$ and $\tau_2(1) = -1$. Then we get the desired output.

---

2. Use the backpropagation algorithm to get an estimate of the derivate of the output of a neural network with respect to the input. Assume for example that the node functions are Lipschitz continuous.

Solution: Suppose that $|\sigma_j(t)| \leq K$ for all j and t. For the backpropagation mtehod one defines $f_j(\mathbf{b}_j) = \mathbf{y}$ and gets

$$f'_{j-1}(\mathbf{b}_{j-1}) = f'_j(\mathbf{b}_j)\sigma'_j(\mathbf{a}_j)W_j.$$

Thus

$$\|f'_{j-1}(\mathbf{b}_{j-1})\| \leq \|f'_j(\mathbf{b}_j)\|K\|W_j\|,$$

and so

$$\|f'_0(\mathbf{b}_0)\| \leq K^L \Pi_{j=1}^L \|W_j\|.$$

Since $\mathbf{b}_0 = \mathbf{x}$ this is what we wanted to calculate.

---

3. Let $f(x) = \frac{1}{1+x^2}$ and $x_j = -5 + \frac{10j}{n}$. Choose the coefficients c_k, $k = 0, \ldots, n$ such that

$$\frac{1}{2}\sum_{j=0}^{n}\left|\sum_{k=0}^{n} c_k x_j^k - f(x_j)\right|^2 + \frac{\lambda}{2}\sum_{k=0}^{n} c_k^2,$$

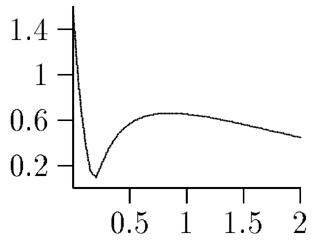
is as small as possible. Investigate numerically how λ should be chosen so that, for example, the error $\sup_{|x|\leq 5}|f(x) - \sum_{k=0}^{n} c_k x^k|$ is as small as possible.

Solution: One can use the following Matlab-program to estimate the error:

```
function [err,c] = apprintp(n,lambda)


x=(-5:10/n:5)';

xt =(-5+5/n:10/n:5-5/n)';

n=size(x,1);
p=n-1;
```

```
A=ones(n,1);
for j=1:p
 A(:,j+1) = A(:,j).*x;
end

y= 1./(1+x.^2);
c = inv(A'*A+lambda*eye(p+1))*A'*y;
c=c(p+1:-1:1);
err = max(abs(polyval(c,xt)-1./(1+xt.^2)));
```

Here one uses a quite crude estimate to find the maximum error. If $n = 10$ one gets the following graph of the error:



C1. Write a Matlab function `makepar` such that `aux=makepar(dim,X,Y)` produces a vector `aux` that contains the dimensions of the network given in `dim`, the input vectors given in the matrix `X` and the outputs in the (matrix) `Y`, so that one easily from `aux` can reproduce the structure, inputs, and outputs of the network. (This vector can then be given as argument to another function such that when it gets the weights and thresholds in another vector, it can calculate everything one wants.)