SELF ORGANIZATION MAPS - SOLUTION TEMPLATE

MAYUR MUDIGONDA

1. Self Organizing Maps

In Figure 1 we see the receptive fields learnt with a small learning rate of .05 and randomly distributed Gaussian blobs all over the input space. Even with a small learning rate sometimes the network converged to funny looking receptive fields (not shown) that do not tile the entire input space accurately. This is largely because of the nature of the objective function and the choice of optimization techniques/algorithms

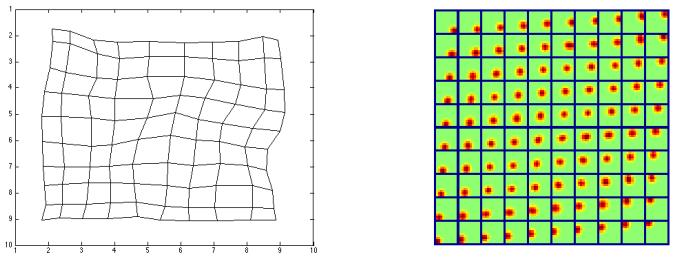

Figure 1. receptive fields that are mapped by self organizing net with Gaussian blobs randomly distributed over the input space

In Figure 2 we see the receptive fields learnt with a small learning rate of .05 and a scotoma in the bottom left of the input space. We can see that the receptive fields adapt and do not learn/represent information there. In Figure 3 we see the receptive fields learnt with a scotoma at the center of the input space. Note how there is a green blob amidst sensitivity in the center receptive fields. Also, note how the input space tiles at the center of the input space to adapt to the scotoma.

In Figure **??** we see the receptive fields learnt with a small learning rate of .05 with over sampling in the upper left quadrant of the input space.

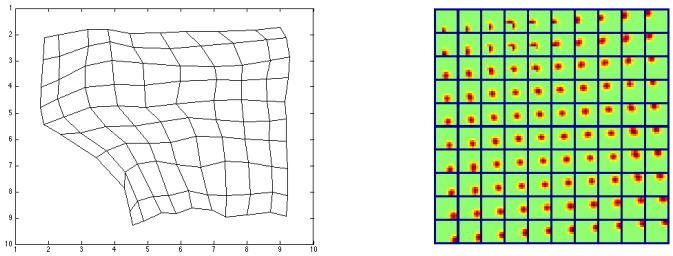

FIGURE 2. receptive fields of a self organizing net with a scotoma at the bottom left of the input space

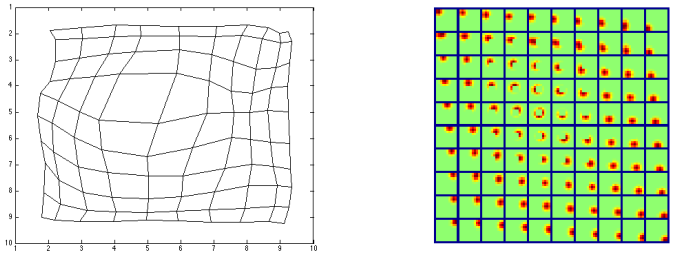

FIGURE 3. receptive fields learnt using a self organizing net with scotoma's in the middle of the input space

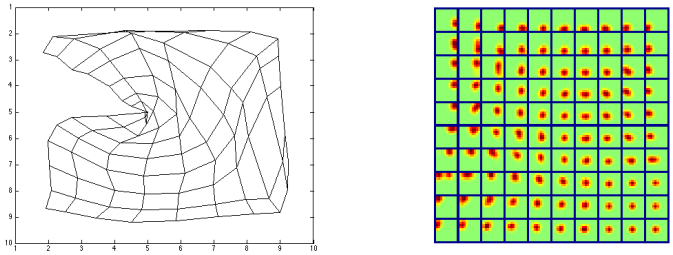

FIGURE 4. receptive fields learnt using a self organizing net with over-sampled input space on the top left part of the input space

2. CODE

2.1. **Problem 1.**

```
% kohonen.m - simulates a self-organizing map
```

```
% image array
imsz=10;
im=zeros(imsz);
[imx imy]=meshgrid(1:imsz,1:imsz); % coordinates in image
showim=0;

% neuron array
SZ=10;
[X Y]=meshgrid(1:SZ,1:SZ);
sigma=1.5;
sigvals = [5 3 1 .5] * sigma;

% weights
W=rand(imsz^2,SZ^2);
W=W*diag(1./sqrt(sum(W.^2))); % normalizes weights

% learning rate
eta=0.05;

figure(1)
%colormap gray
%subplot(121)
%h=imagesc(im,[0 1]);
%axis image

for sigma = sigvals
    t=0;
    while (t<1000)

      % paint a Gaussian blob at random position in image
        x=ceil(imsz*rand);
        y=ceil(imsz*rand);
        im=exp(-0.5*((x-imx).^2+(y-imy).^2));
        %set(h,'CData',im)
        %drawnow

      % compute output and do wta
        Z = W'*im(:);
        [Zmax,winner] = max(Z);
        winner = winner(1);   just to be safe -- if there are two identical outputs
        Xwin = X(winner);
        Ywin = Y(winner);
```

```
      % spread activation to neighbors
        lambda = exp(-0.5*((Xwin-X).^2+(Ywin-Y).^2)/(sigma^2));

      % Hebbian weight update
      %spreading activation across a neighborhood across all neurons
        dW = eta*(repmat(im(:),1,SZ^2)-W)*diag(lambda(:));
        W = W+dW;
        W=W*diag(1./sqrt(sum(W.^2))); % normalize weight vectors

      % display network every 100 trials
      if mod(t,100)==0
        %subplot(122)
        mux=reshape((abs(W)'*imx(:))./sum(abs(W))',SZ,SZ);
        muy=reshape((abs(W)'*imy(:))./sum(abs(W))',SZ,SZ);
        plot(mux,muy,'k'), hold on
        plot(mux',muy','k'), hold off
        axis ij, axis([1 SZ 1 SZ])
        drawnow
        figure(2)
        showrfs(W)
        figure(1)
      end

      t=t+1;
    end
end
```

2.2. **Problem 2.**

```
% kohonen.m - simulates a self-organizing map

% image array
imsz=10;
im=zeros(imsz);
[imx imy]=meshgrid(1:imsz,1:imsz); % coordinates in image
showim=0;

% define scotoma
scotx = 4:6;
scoty = 4:6;
% scotoma = find(ismember(imx,scotx) & ismember(imy,scoty));
% neuron array
```

```
SZ=10;
[X Y]=meshgrid(1:SZ,1:SZ);
sigma=1.5;
sigvals = [5 3 1 .5] * sigma;

% weights
W=rand(imsz^2,SZ^2);
W=W*diag(1./sqrt(sum(W.^2))); % normalizes weights

% learning rate
eta=0.1;

figure(1)
%colormap gray
%subplot(121)
%h=imagesc(im,[0 1]);
%axis image

for sigma = sigvals
    t=0;
    while (t<2000)

      % paint a Gaussian blob at random position in image, but with dead
      % pixels in scotoma
        x=ceil(imsz*rand);
        y=ceil(imsz*rand);
        im=exp(-0.5*((x-imx).^2+(y-imy).^2));
        im(scotx,scoty) = 0;
        %set(h,'CData',im)
        %drawnow

      % compute output and do wta
        Z = W'*im(:);
        [Zmax,winner] = max(Z);
        winner = winner(1);  % just to be safe -- if there are two identical outputs, take the
        Xwin = X(winner);
        Ywin = Y(winner);

      % spread activation to neighbors
        lambda = exp(-0.5*((Xwin-X).^2+(Ywin-Y).^2)/(sigma^2));

      % Hebbian weight update
        dW = eta*(repmat(im(:),1,SZ^2)-W)*diag(lambda(:));
```

```
        W = W+dW;
        W=W*diag(1./sqrt(sum(W.^2))); % normalize weight vectors

      % display network every 100 trials
      if mod(t,100)==0
        %subplot(122)
        mux=reshape((abs(W)'*imx(:))./sum(abs(W))',SZ,SZ);
        muy=reshape((abs(W)'*imy(:))./sum(abs(W))',SZ,SZ);
        plot(mux,muy,'k'), hold on
        plot(mux',muy','k'), hold off
        axis ij, axis([1 SZ 1 SZ])
        drawnow
        figure(2)
        showrfs(W)
        figure(1)
      end

      t=t+1;
    end
end
```

2.3. **Problem 3.**

```
% kohonen.m - simulates a self-organizing map

% image array
imsz=10;
im=zeros(imsz);
[imx imy]=meshgrid(1:imsz,1:imsz); % coordinates in image
showim=0;

% define area of overstimulation
overstimx = 5:6;
overstimy = 5:6;
overstimp = .3;

% neuron array
SZ=10;
[X Y]=meshgrid(1:SZ,1:SZ);
sigma=1.5;
sigvals = [5 3 1 .5] * sigma;

% weights
W=rand(imsz^2,SZ^2);
```

```
W=W*diag(1./sqrt(sum(W.^2))); % normalizes weights

% learning rate
eta=0.05;

figure(2)
title('Receptive fields');

figure(1)
title('Network mapping');
%colormap gray
%subplot(121)
%h=imagesc(im,[0 1]);
%axis image

for sigma = sigvals
    t=0;
    while (t<2000)

      % paint a Gaussian blob at random position in image, but with
      % higher probability of landing in one area.
      if rand > overstimp   % normally, just distribute evenly
        x=ceil(imsz*rand);
        y=ceil(imsz*rand);
      else                  % some of the time, only land in one area
        x = floor(range(overstimx)*rand) + min(overstimx);
        y = floor(range(overstimy)*rand) + min(overstimy);
      end
        im=exp(-0.5*((x-imx).^2+(y-imy).^2));
        %set(h,'CData',im)
        %drawnow

      % compute output and do wta
        Z = W'*im(:);
        [Zmax,winner] = max(Z);
        winner = winner(1);  % just to be safe -- if there are two identical outputs, take the
        Xwin = X(winner);
        Ywin = Y(winner);

      % spread activation to neighbors
        lambda = exp(-0.5*((Xwin-X).^2+(Ywin-Y).^2)/(sigma^2));

      % Hebbian weight update
```

```
        dW = eta*(repmat(im(:),1,SZ^2)-W)*diag(lambda(:));
        W = W+dW;
        W=W*diag(1./sqrt(sum(W.^2))); % normalize weight vectors

      % display network every 100 trials
      if mod(t,100)==0
        %subplot(122)
        mux=reshape((abs(W)'*imx(:))./sum(abs(W))',SZ,SZ);
        muy=reshape((abs(W)'*imy(:))./sum(abs(W))',SZ,SZ);
        plot(mux,muy,'k'), hold on
        plot(mux',muy','k'), hold off
        axis ij, axis([1 SZ 1 SZ])
        drawnow
        figure(2)
        showrfs(W)
        figure(1)
      end

      t=t+1;
    end
end
```