EN.520.665. HW3

1. for GMRF models

$$y(s) = \sum_{r\in N_a} \theta_r (y(s+r) + y(s-r)) + e(s)$$

For the first order case $N = \{(0,-1), (0,1), (-1,0), (1,0)\}$

$$N_s = \{(0,1), (1,0)\}$$

when Assume the doubly-periodic boundary conditions

$$y(s) = \sum_{r\in N} \theta_r (s \bmod r) + e(s)$$

$$(s \bmod r) = ([(s_1+r_1) \bmod M, (s_2+r_2) \bmod M])$$

$$E(e(s)e(r)) = \begin{cases} -\theta_{r-s} v & \text{if } (s-r) \bmod M \in N \\ v & \text{if } s=r \bmod M \\ 0 & \text{otherwise} \end{cases}$$

when $v > 0$ $\quad B(\theta) y = e$

$$y = [y(0,0), \cdots y(0,M-1), y(1,0), \cdots, y(1,M-1), \cdots y(M-1,M-1)]^T$$

$$e = [e(0,0), \cdots, e(0,M-1), e(1,0), \cdots; e(1,M-1), \cdots, e(M-1,M-1)]^T$$

$B(\theta)$ matrix will be

$$B(\theta) = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{12} \\ B_{12} & B_{11} & \ddots & \vdots \\ \vdots & \vdots & & \\ B_{12} & B_{13} & \cdots & B_{11} \end{bmatrix}$$

$$E\{ee^T\} = v B(\theta)$$

$$E\{y e^T\} = vI$$

$$E\{yy^T\} = v[B(\theta)]^{-1}$$

In order for the MRF stable, the matrix must be positive definite

$\therefore$ $(k_1, k_2) \in K = \{k; 0 \le k_1, k_2 \le M-1\}$ also greater than 0

$$\mu_k = 1 - \theta^T \phi_k > 0, \quad k \in K$$

$$\phi_k = [\exp(\frac{j2\pi k^T r}{M}); r \in N]^T$$

$$v_k = 1 - \theta^T \phi_k$$

2. for Gibbs Sampling algorithm: f be a random configuration

For $i \in S$, compute $P_l = P(f_i = l \mid f_{N_i}) \forall l \in \mathcal{L}$

and set f_i to l with probability P

In the Metropolis Sampling: We will first Randomly initialize f and define image lattice S

For $i \in S$: We first let $f'_{i'} = f_{i'}$, for all $i' \neq i$

the choose $f_i \in \mathcal{L}$ at random and Let $P = \min\{1, \frac{P(f')}{P(f)}\}$, After that replace f by f' with probability P and generate a uniform random variable $u \in (0,1)$.

Finally judge if $u \in P$ replace f by f'

if $u > P$ then no change.

When appling to a $N \times N$ texture image, by virtue no less than one of the unforeseen scattering can't be supportively inspected then the Gibbs sampling is not suitable. Also if the model are non linear in the limits, the limit prohibitive maybe not known. So we need a capable system Metropolis-Hasting computation. Doing the Metropolis-Hastings requires simply drawing from the suggestion, drawing a uniform discretionary variable and evaluating the affirmation measure. For Gibbs, it have a quite low efficient that limits are connected considering the way that you can't take to one side steps. So, the Metropolis Sampling is a better choice.

3. Alexnet: The Alexnet has eight layers with learnable parameters. The model consist of five layers with a combination of max pooling followed by 3 fully connected layers. The basic computation is about 660 K units, 61 M parameters and over 600 M connections.

computation example :

conv1 : $(11\times11)\times3\times96+96 = 34944$
conv2 : $(5\times5)\times96\times256+256 = 614656$
conv3 : $(3\times3)\times256\times384+384 = 885120$
conv4 : $(3\times3)\times384\times384+384 = 1327488$
conv5 : $(3\times3)\times384\times256+256 = 884992$
fc1 : $(6\times6)\times256\times4096+4096 = 37752832$
fc2 : $4096 * 4096+4096 = 16781312$
fc3 : $4096\times1000+1000 = 4097000$ → total = 62378344
near 61 M

VGGnet : VGGnet 16 has a total of 138 million parameters.

conv3-64 x 2 : 38720
conv3-128 x 2 : 221440
conv3-256 x 3 : 1475328
conv3-512 x 3 : 5899776
conv3-512 x 3 : 7079424
fc1 : 102,764,544
fc2 : 16781,312
fc3 : 4097000
Total : 138357544

Resnet: Resnet-50 has over 23 million parameters,
Resnet-18 has 11 million trainable parameters :
I can't calculate 50 layer parameters, but I can run code to get ;

```
resnet_model = Resnet50
    (weights=" ")
resnet_model.summary()
```

Inception net : each inception module consists of four operations in parrallel
1x1 conv
3x3 conv
5x5 conv
max pooling

for example inception (3a)

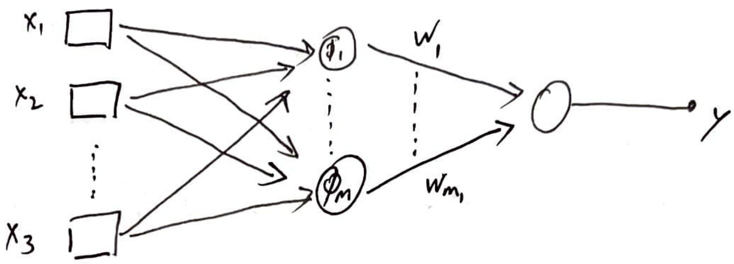
conv1x1a : 18528
conv1x1b : 3088
maxpool-a : 0
: 12352
conv1x1c : 110720
conv3-3 : 12832
conv5x5 : 6176
conv1x1d

4. The transfer function of a hidden units is linear. A three layer N/W is equivalent to a ~~two~~ - layer one.

First set the function : inputlayer : source of node the connect the Network with it's environment.

Hidden layer : Apply a Non-linear transfunction from the input spot to the hidden spot.

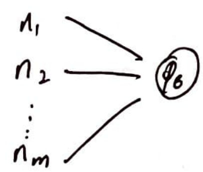
output layer : Apply a linear transformation from the hidden spon to the output spot.



as the picture shows, a three layer N/W is equivalent to a two-layer one.

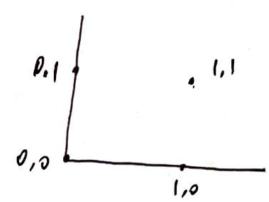
a function is approximated as a linear combination of radial basis function when use a radial basis-function in Hidden units.

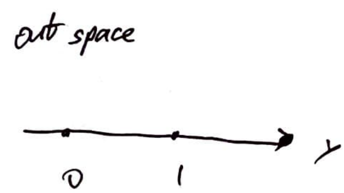
$\phi_6 (\| x - t \|^2)$ the output depends on the distance of the input from the contract.



~~大~~ large spread → less sensitivity

In the XOR problem





we construct an RBF pattern classificial

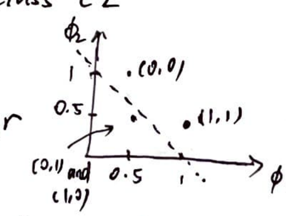
(0,0) and (1,1) are mapped to 0, class C1

(1,0) and (0,1) are mapped to 1, class C2

in the hidden space

$\phi_1 (n_1, n_2) = e^{-\| x - t_1 \|^2}$ $t_1 = [1,1]^r$

$\phi_2 (n_1, n_2) = e^{-\| x - t_2 \|^2}$ $t_2 = [0,0]^r$



when mapped into the feature space (ϕ_1, ϕ_2), C_1 and C_2 become linearly sperably, so a linear class with $\phi_1(n)$ and $\phi_2(n)$ as input can be used to solve XOR problem

∴ a three-layer network with linear hidden units cannot solve a non-linearly seperable problem such as XOR

5. A learning rate that is too huge can make the model join excessively fast to an imperfect arrangement. We can detect it by analysing the cost function value. If the cost function value is too high and it keeps on increasing with iteration which shows that learning rate is too high.

when a learning rate is too low, it will be too little to make the interaction stall out. The preparing will advance gradually as you are making extremely minuscule updates to the loads in your organization. We can detect it by analysing the cost function value. If the cost function value is low, the learning rate is low.