Homework #4

4.1 More about VC Dimension

(1) We apply an exponentiation to both side of the inequality

$$2^{\mathrm{VCD(G)}} \leq 2^{\log_2 \mathrm{L}} = \mathrm{L} \tag{4.1.1}$$

Which mean the number of all patterns we may generate is less or equal to total number of classifier we have. It is satisfy by the truth of the relation between classifiers and patterns are one to one mapping.

(2) We apply an exponentiation to both side of the first part of inequality

$$1 = 2^0 \leq 2^{\mathrm{VCD}(\mathrm{G}_1 \cap \mathrm{G}_2)} = 2^{\mathrm{VCD}(\mathrm{G}')} \tag{4.1.2}$$

It is trivial satisfy by the fact of we can at least generate one pattern if we have a classifier, say G'. And we see the second part of the inequality

$$\mathrm{VCD}(\mathrm{G}') = \mathrm{VCD}(\mathrm{G}_1 \cap \mathrm{G}_2) \leq \min(\mathrm{D}_1, \mathrm{D}_2) \tag{4.1.3}$$

G' is the intersection of G_1 and G_2, It is satisfy by the set theory, the maximum of $\mathrm{VCD}(\mathrm{G}')$ is exactly D_1 or D_2 set, depend on which one is smaller.

(3) First part of the inequality also use the set theory, we proof that the first part is satisfy by the fact the minimum of $\mathrm{VCD}(\mathrm{G}'')$, where, G'' is the union of G_1 and G_2, is equal to D_1 or D_2 set, depend on which one is bigger. Therefore,

$$\max(\mathrm{D}_1, \mathrm{D}_2) \leq \mathrm{VCD}(\mathrm{G}'') = \mathrm{VCD}(\mathrm{G}_1 \cup \mathrm{G}_2) \tag{4.1.4}$$

For second part, assume that the upper bound is $\mathrm{D}_1 + \mathrm{D}_2 + 2$. We can separate the bits in two parts, one has $\mathrm{D}_1 + 1$ bits, and another one has $\mathrm{D}_2 + 1$ bits. If using G1, the patterns is less than equal to $\mathrm{B}(\mathrm{D}_1 + \mathrm{D}_2 + 2, \mathrm{D}_2 + 1)$. In the other hand, if using G2, the patterns is less than equal to $\mathrm{B}(\mathrm{D}_1 + \mathrm{D}_2 + 2, \mathrm{D}_1 + 1)$. The fact maximum numbers of pattern is

$$\mathrm{B}(\mathrm{D}_1 + \mathrm{D}_2 + 2, \mathrm{D}_2 + 1) + \mathrm{B}(\mathrm{D}_1 + \mathrm{D}_2 + 2, \mathrm{D}_1 + 1) =$$

$$2^{\mathrm{D}_1 + \mathrm{D}_2 + 2} - \binom{\mathrm{D}_1 + \mathrm{D}_2 + 1}{\mathrm{D}_1 + 2} - \binom{\mathrm{D}_1 + \mathrm{D}_2 + 1}{\mathrm{D}_2 + 1} \tag{4.1.5}$$

Which is smaller than $2^{\mathrm{D}_1 + \mathrm{D}_2 + 2}$.We find out that each part cannot generate all pattern because the constraint of VCD. Because the $\mathrm{VCD} \in \mathrm{N}$, the upper bound will shrink to $\mathrm{D}_1 + \mathrm{D}_2 + 1$.

4.2 Curse of Dimensionality

(1) Consider about the volume of hypercube $[-0.5,0.5]^d$with d dimension can be written by

$$V_c^d = (0.5 + 0.5)^d = 1 \tag{4.2.1}$$

A hypersphere of radius R in d-dimensions can be represent by a d-tuples of points such that

$$x_1{}^2 + x_2{}^2 + \cdots + x_d{}^2 = R^2 \tag{4.2.2}$$

Let V_s^d denote the `content' of a d-hypersphere or radius R is given by

$$V_s^d = \int_0^R S_d r^{d-1} * dr = \frac{S_d R^d}{d} \tag{4.2.3}$$

By integration in n-d spherical coordinates (Stewart, 2006), we have a recurrence relation:

$$V_s^d = \frac{2\pi R^2}{n} V_s^{d-2} \tag{4.2.4}$$

Thus, by define $V_s^0 = 1$ and $V_s^1 = 0.2$ the log ratios $\log(V_s^d/V_c^d)$ for d = 1, 2 … 10 can be showed by figure 4.2(1)

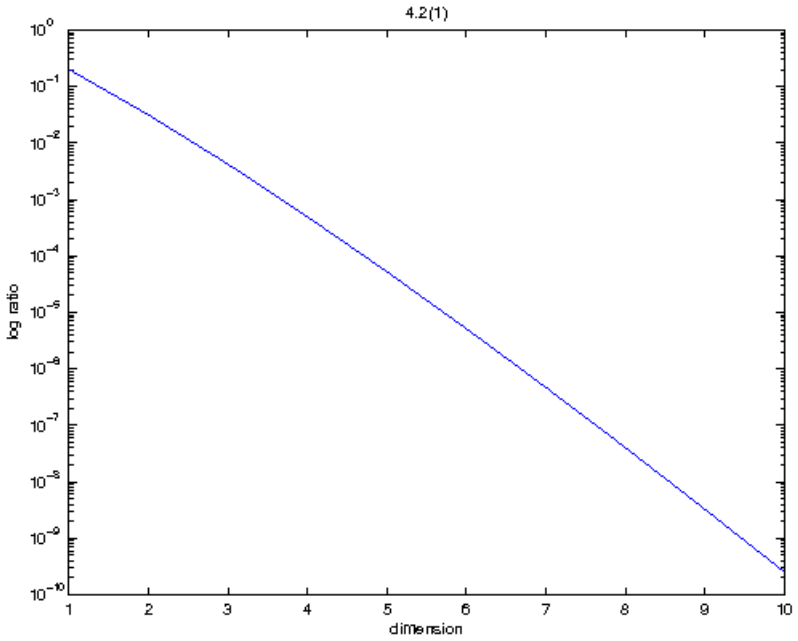


Figure 4.2(1)

And the ratios are show in Table 4.2(1)

d = 1	0.200000000000000	d = 6	0.000005167712780
d = 2	0.031415926535898	d = 7	0.000000472476597
d = 3	0.004188790204786	d = 8	0.000000040587121
d = 4	0.000493480220054	d = 9	0.000000003298509
d = 5	0.000052637890139	d = 10	0.000000000255016

Table 4.2(1)

(2) We can reduce this problem to problem 4.2(1). Because we generate the points uniformly in the hypercube. The probability of origin's nearest neighbor in the training set within 0.1 can approach to the ratio V_s^d/V_c^d we compute above. Hence we want to ensure the probability $\geq \frac{1}{2}$, where the probability

$$1-\left(1-\frac{V_s^d}{V_c^d}\right)^n \geq \frac{1}{2} \tag{4.2.5}$$

Therefore, by inequality $(4.2.5)$

$$n \geq \log\left(\frac{1}{2}-\left(1-\frac{V_s^d}{V_c^d}\right)\right) \tag{4.2.6}$$

We show n in Table 4.2(2),

$d = 1$	4	$d = 6$	134131
$d = 2$	22	$d = 7$	1467051
$d = 3$	166	$d = 8$	17078008
$d = 4$	1405	$d = 9$	210139551
$d = 5$	13168	$d = 10$	2718048871

Table 4.2(2)

(3) We can find out the volumes of d-hypersphere inside the hypercube is decrease by exponentiation with base 2, when we move it to a corner of hypercube. We have the new statement

$$1-\left(1-\left(\frac{1}{2}\right)^d\frac{V_s^d}{V_c^d}\right)^n \geq \frac{1}{2} \tag{4.2.7}$$

Therefore, by inequality $(4.2.7)$

$$n \geq \log\left(\frac{1}{2}-\left(1-\left(\frac{1}{2}\right)^d\frac{V_s^d}{V_c^d}\right)\right) \tag{4.2.8}$$

We have Table 4.2(3)

$d = 1$	7	$d = 6$	8584343
$d = 2$	88	$d = 7$	187782504
$d = 3$	1324	$d = 8$	4371971194
$d = 4$	22474	$d = 9$	107591417388
$d = 5$	421383	$d = 10$	2783466236365

Table 4.2(3)

The finding is we may suffer by the dimension of hypercube. For example, we need lots of sample to ensure the data similarity is consisted. When we try to query the point in the corner of testing data, we need exponentiation time sample to achieve that goal.

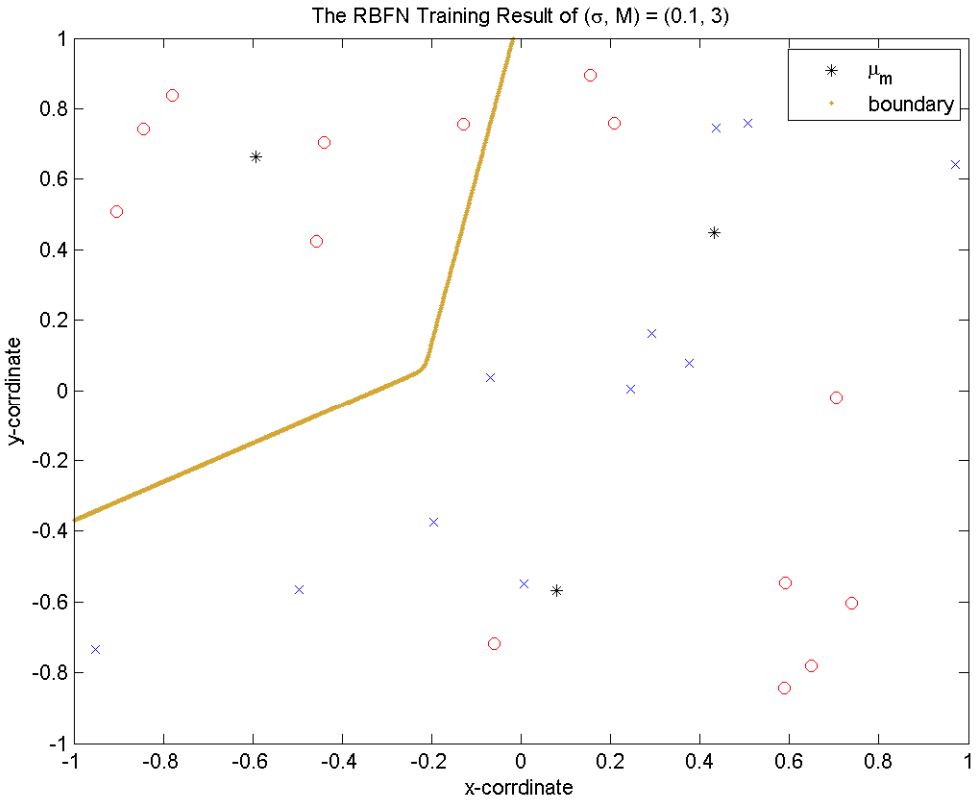
4.3 Experiment with Radial Basis Function Network

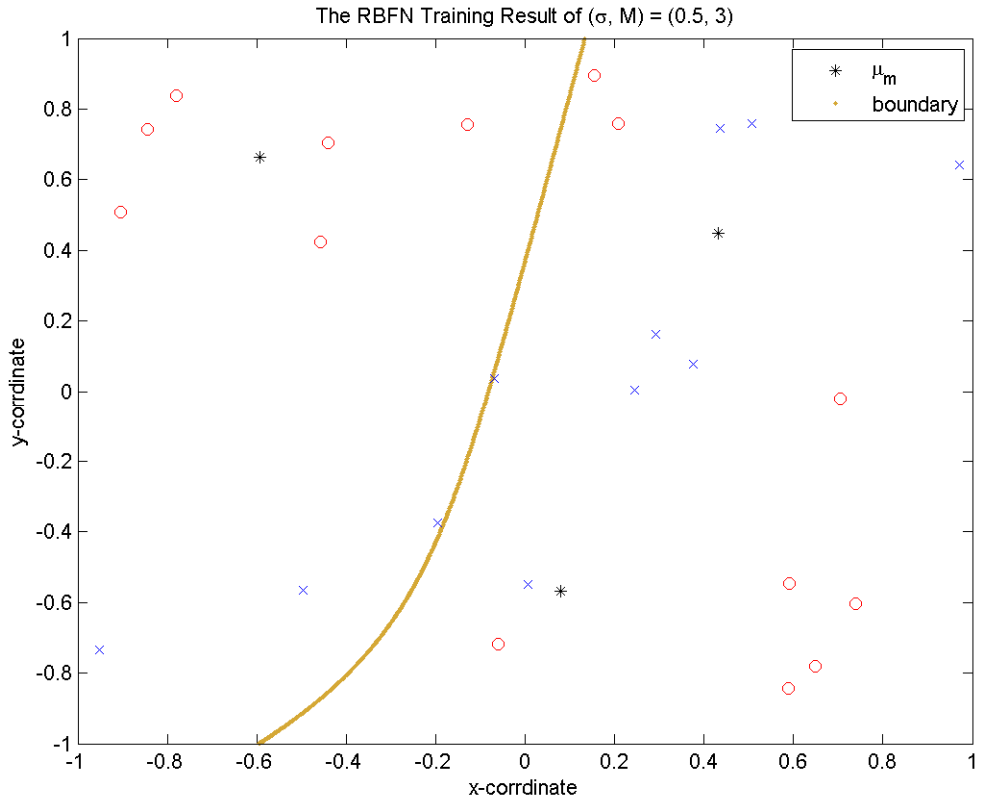
(1) Some brief finding

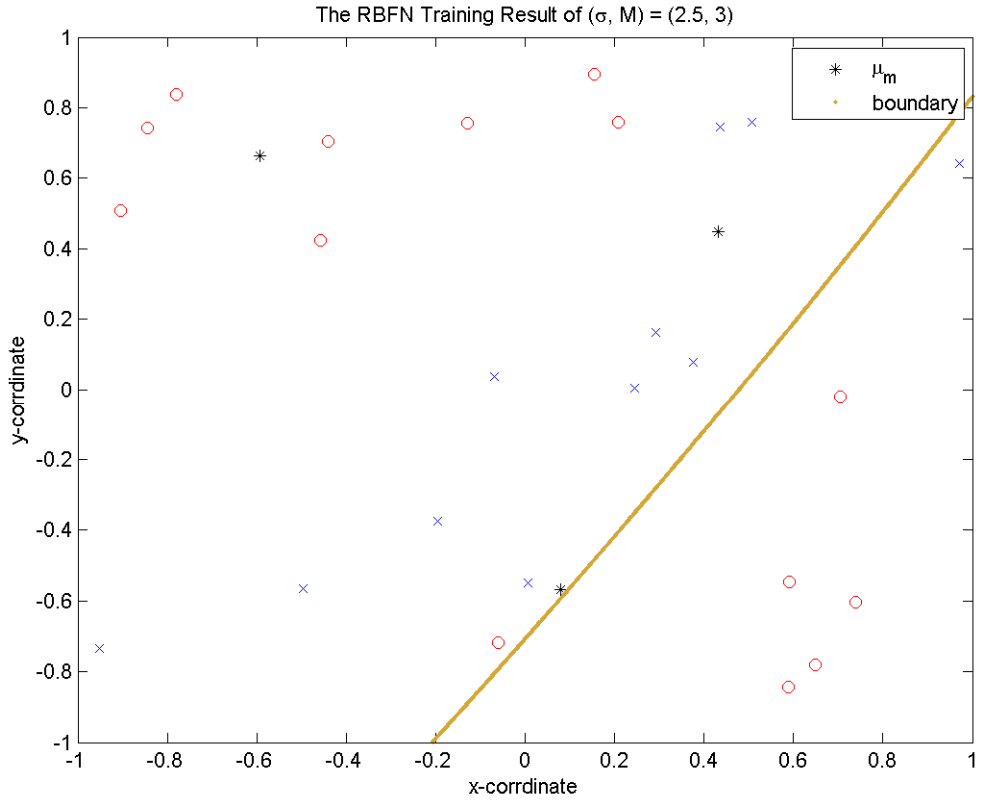
(a) When M too small, we cannot get a good result on Radial Basis Function Network.

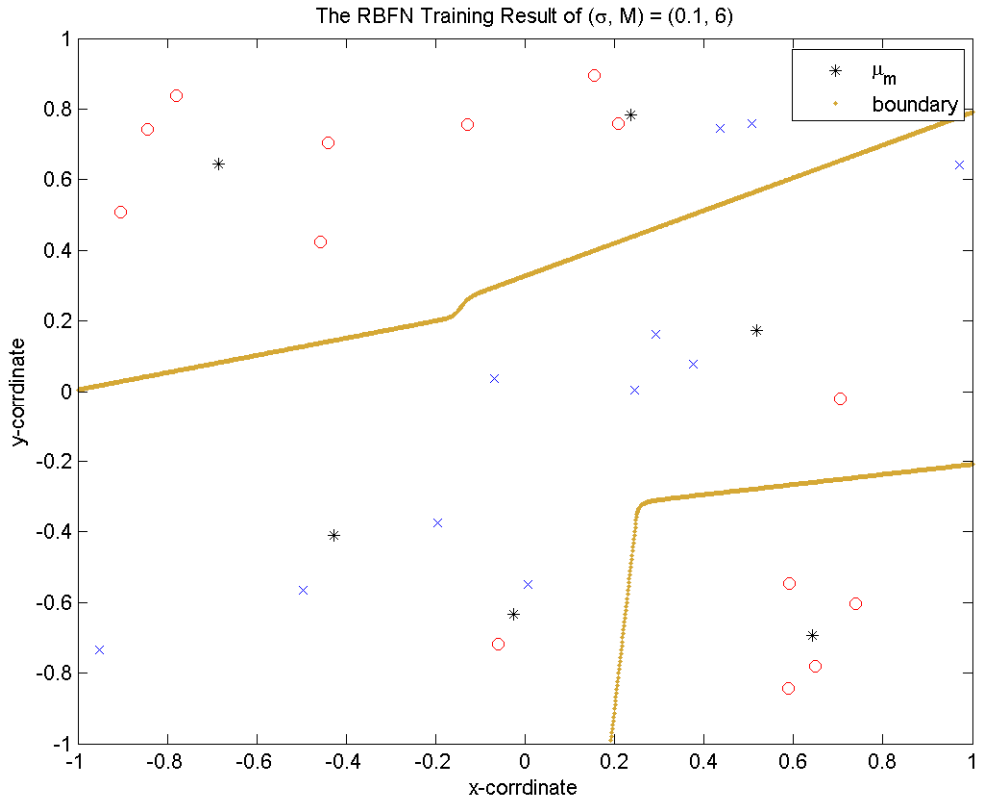
(b) When delta value gets bigger, we will get a smooth boundary.

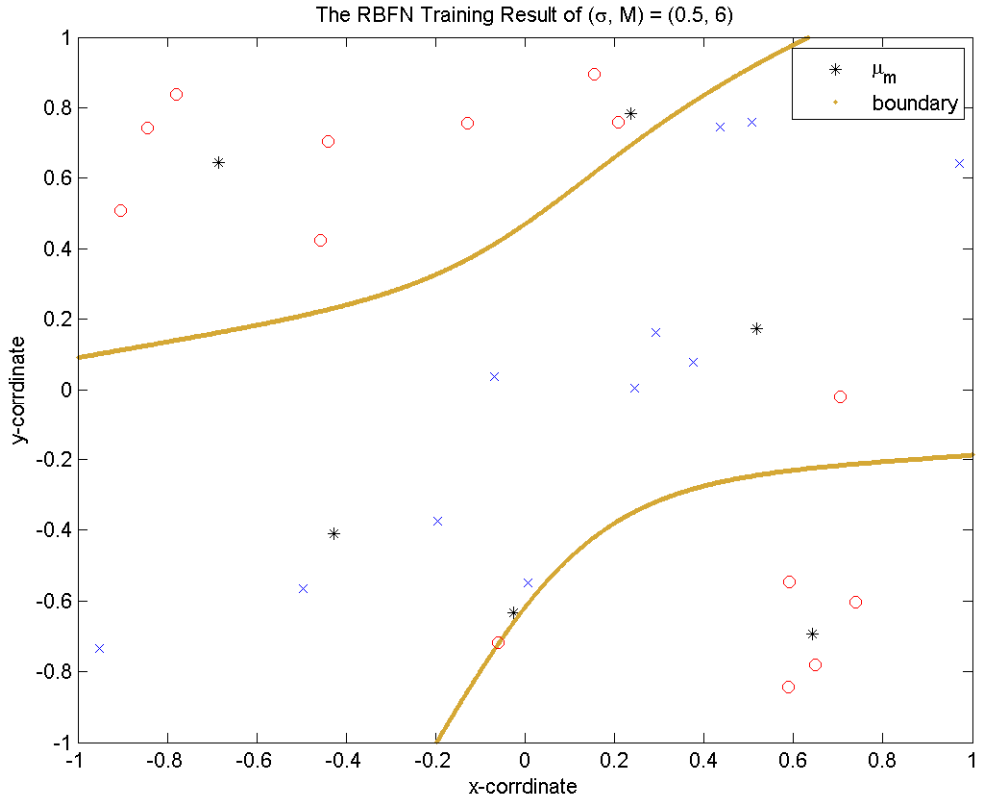
The figures show below

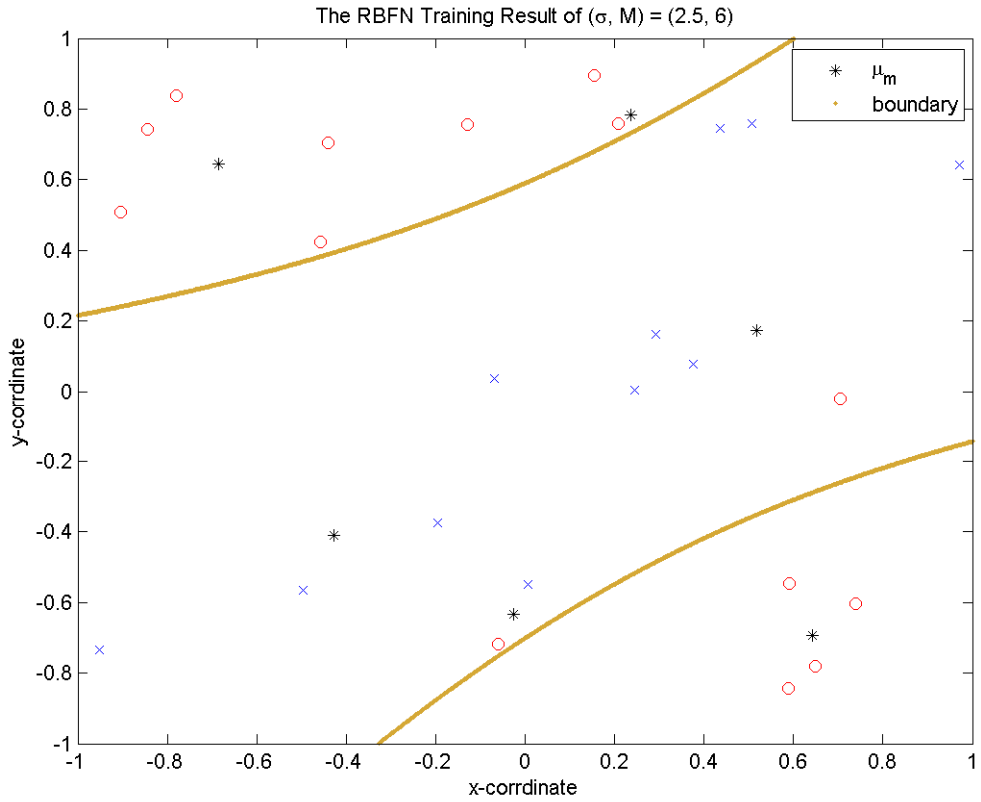


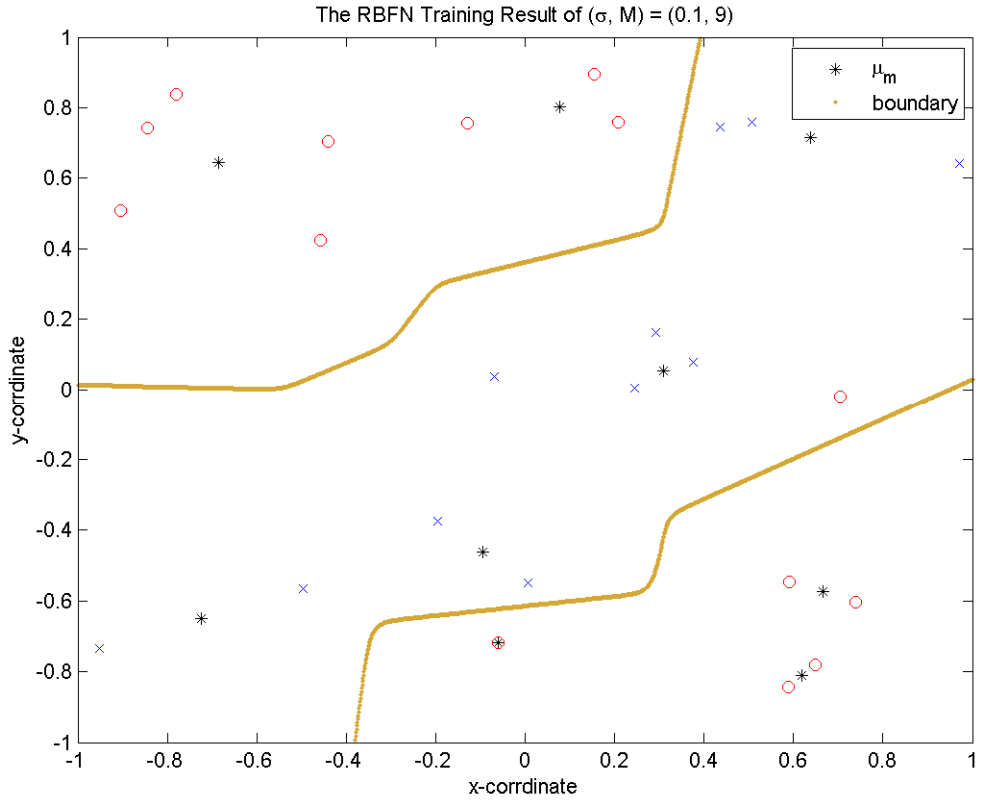

The RBFN Training Result of (σ, M) = (0.5, 3)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

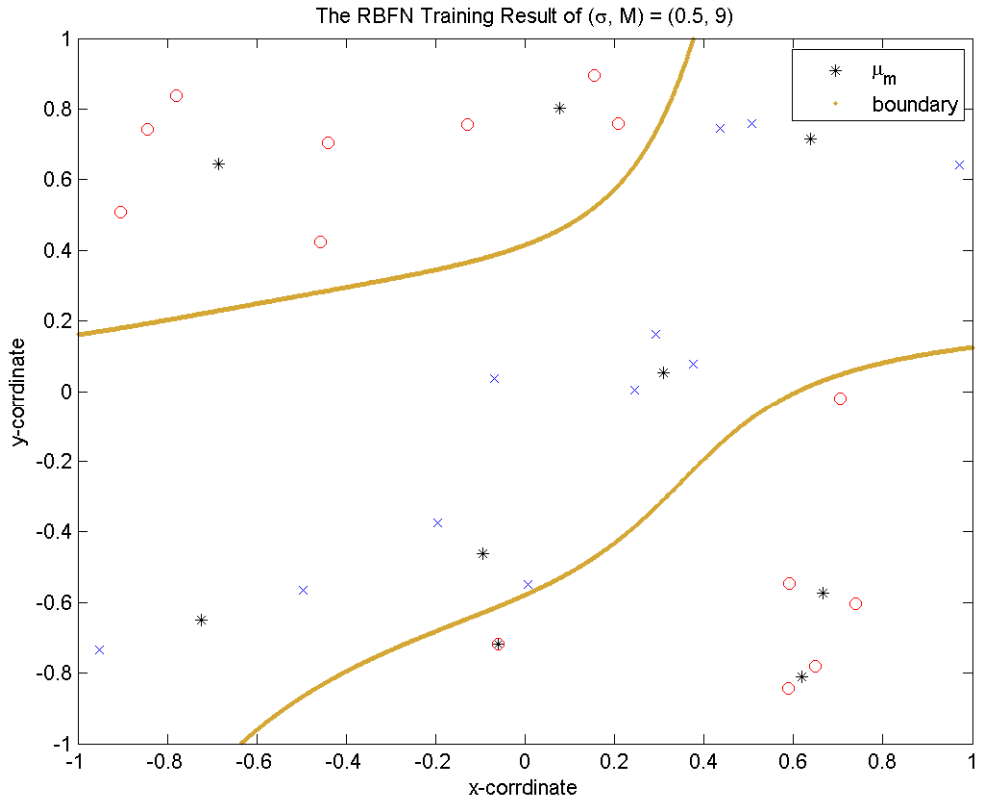

The RBFN Training Result of (σ, M) = (2.5, 3)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

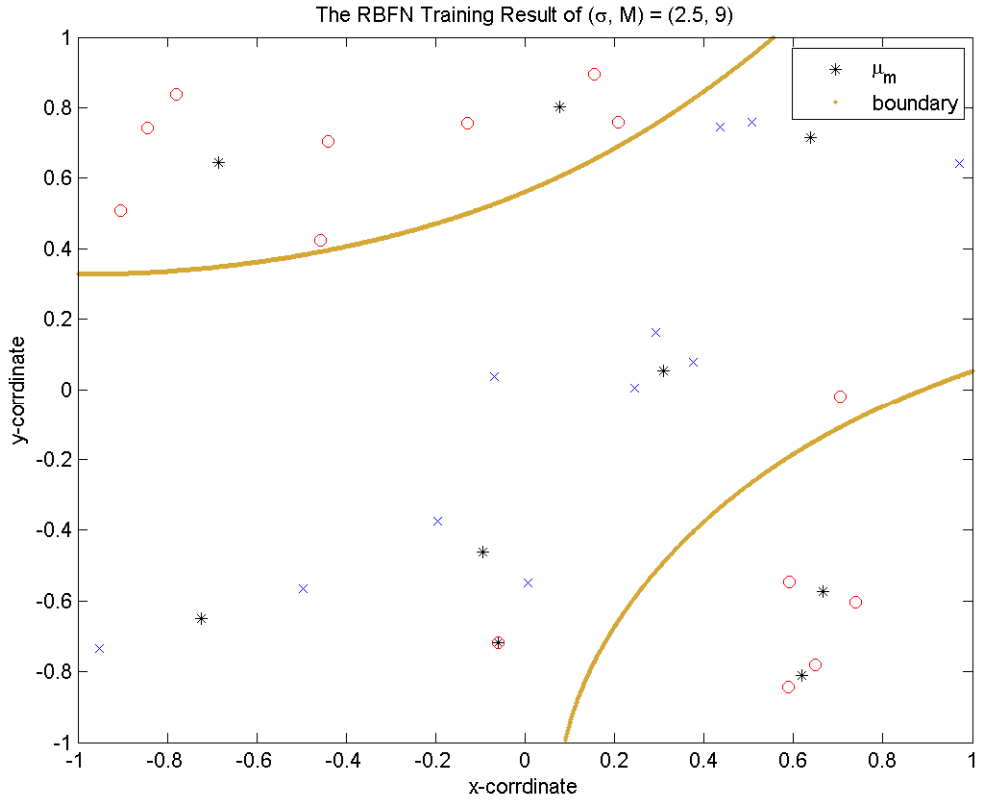

The RBFN Training Result of (σ, M) = (0.1, 6)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


The RBFN Training Result of (σ, M) = (0.5, 6)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


The RBFN Training Result of (σ, M) = (2.5, 6)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


The RBFN Training Result of (σ, M) = (0.1, 9)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


The RBFN Training Result of (σ, M) = (0.5, 9)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


The RBFN Training Result of (σ, M) = (2.5, 9)
μm
boundary
y-corrdinate
x-corrdinate
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

(2) The values show below

ν	$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 2.5$
$M = 3$	0.28	0.28	0.60
$M = 6$	0.16	0.08	0.08
$M = 9$	0.20	0.04	0.08

$\hat{\pi}$	$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 2.5$
$M = 3$	0.356	0.340	0.552
$M = 6$	0.212	0.104	0.180
$M = 9$	0.464	0.216	0.180

We get the best result on $\sigma = 0.5$ with $M = 6$, which is not have a best training error. The fact imply one thing, those get lower training error one may over fitting the training data, like $\sigma = 0.5$ with $M = 9$.

4.4 Experiment with Backprop Neural Network

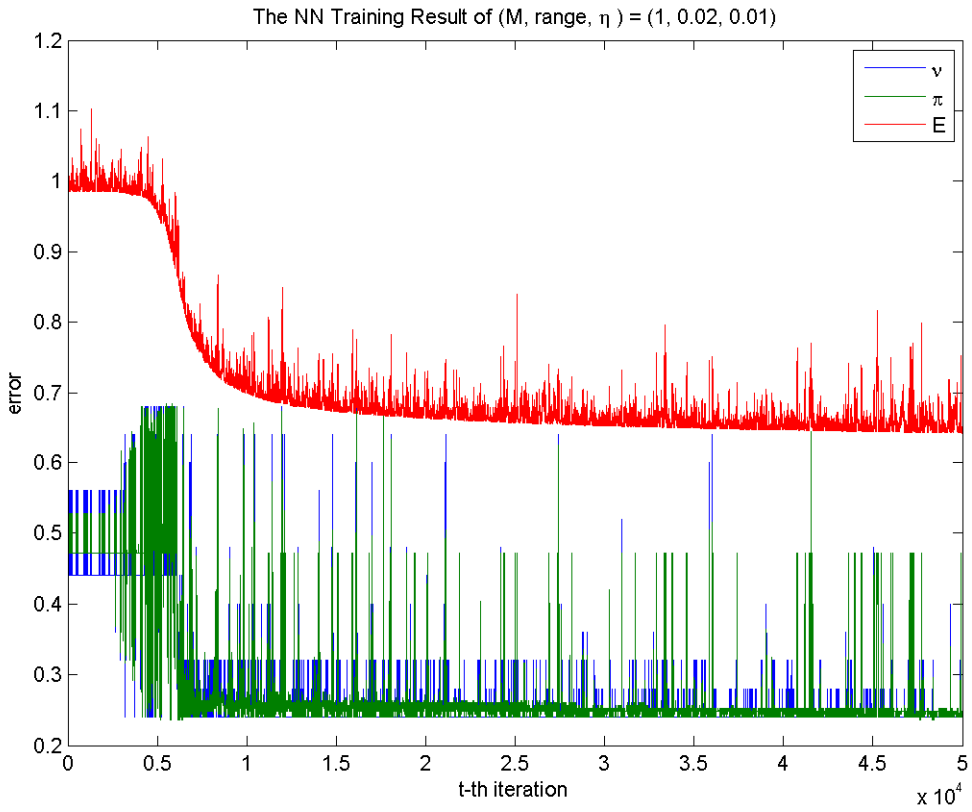
(1) The conclusion is

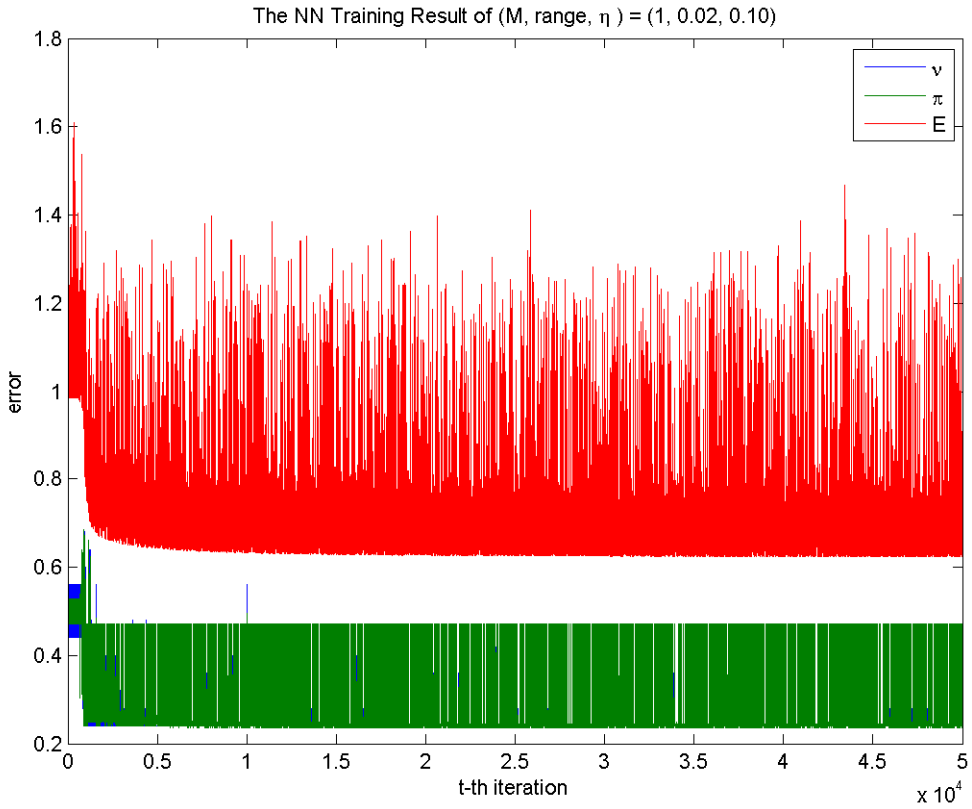
(a) With a large learning rate, the E value cannot converge in a good place, which means we get a bad result in every case with big learning rate.

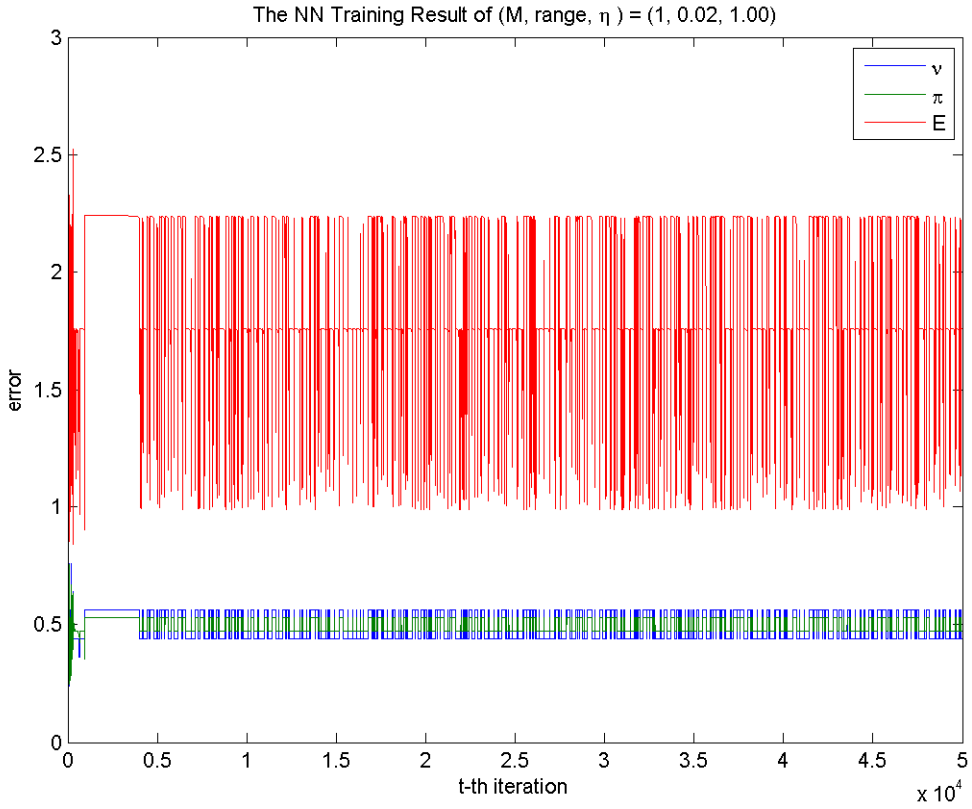
(b) With a proper M, say 3, we can get a good result with a lower E value.

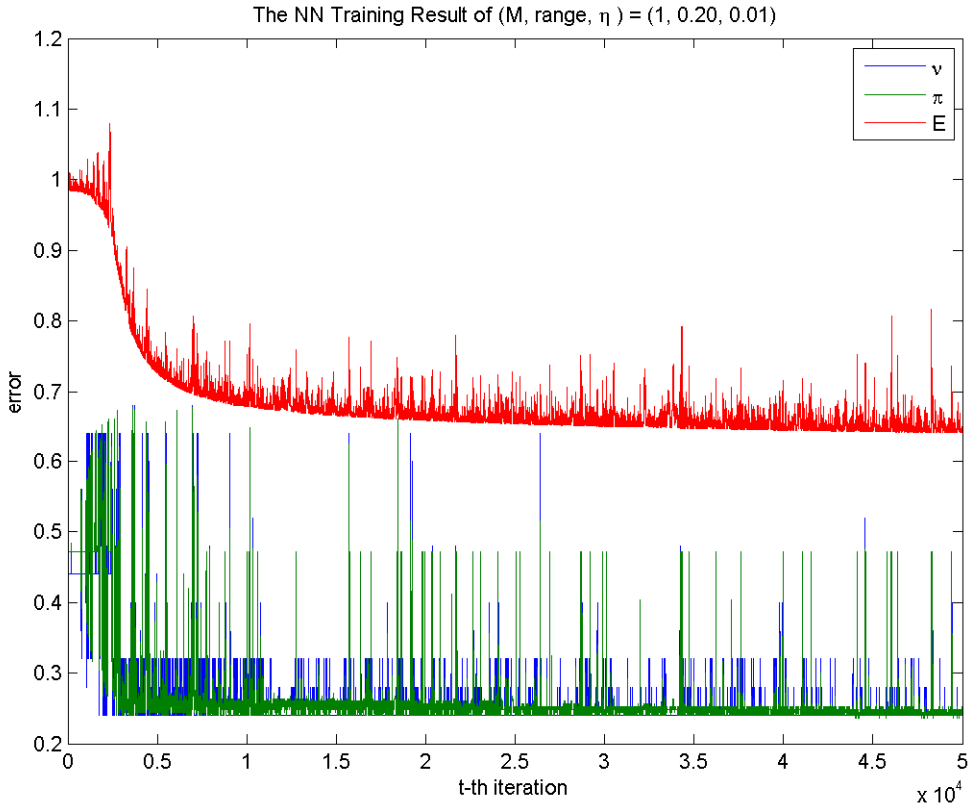
(c) The random initial value looks it not that important in our case.

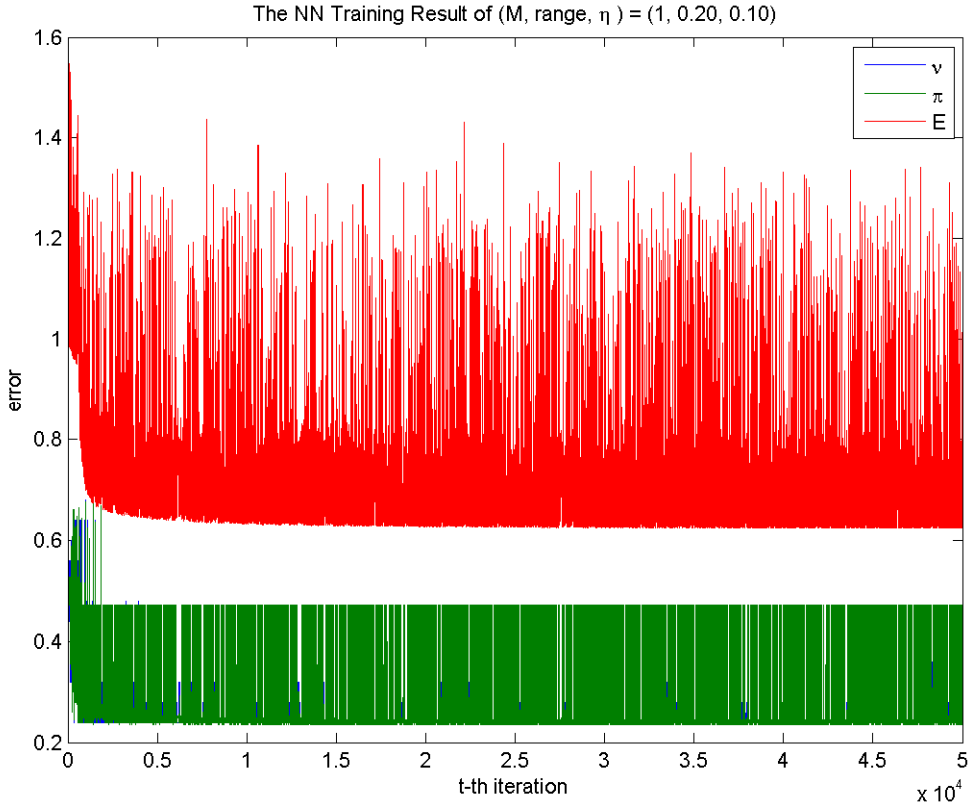
The figures show as below:

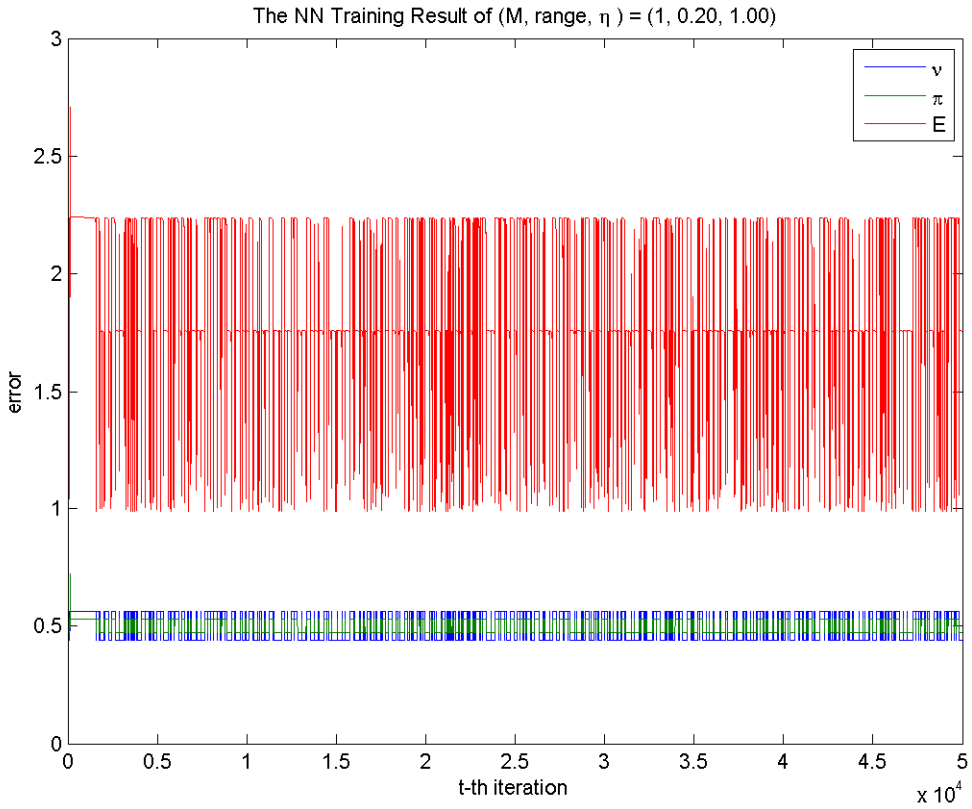


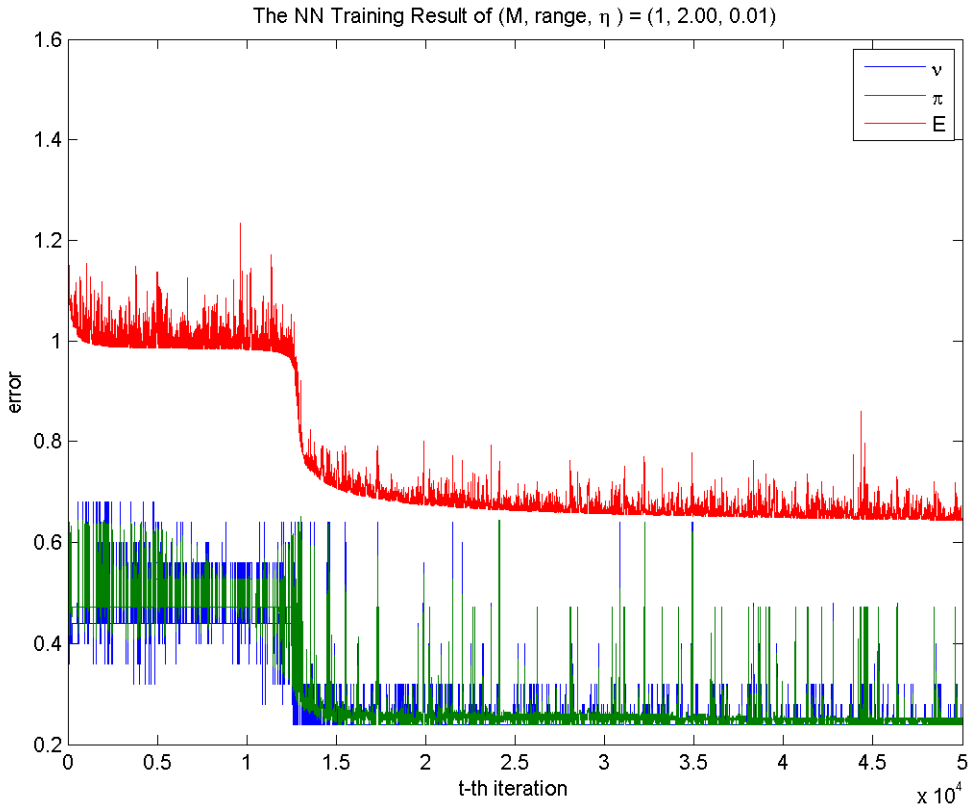

The NN Training Result of (M, range, η) = (1, 0.02, 0.10)
ν
π
E
error
t-th iteration
x 10^4

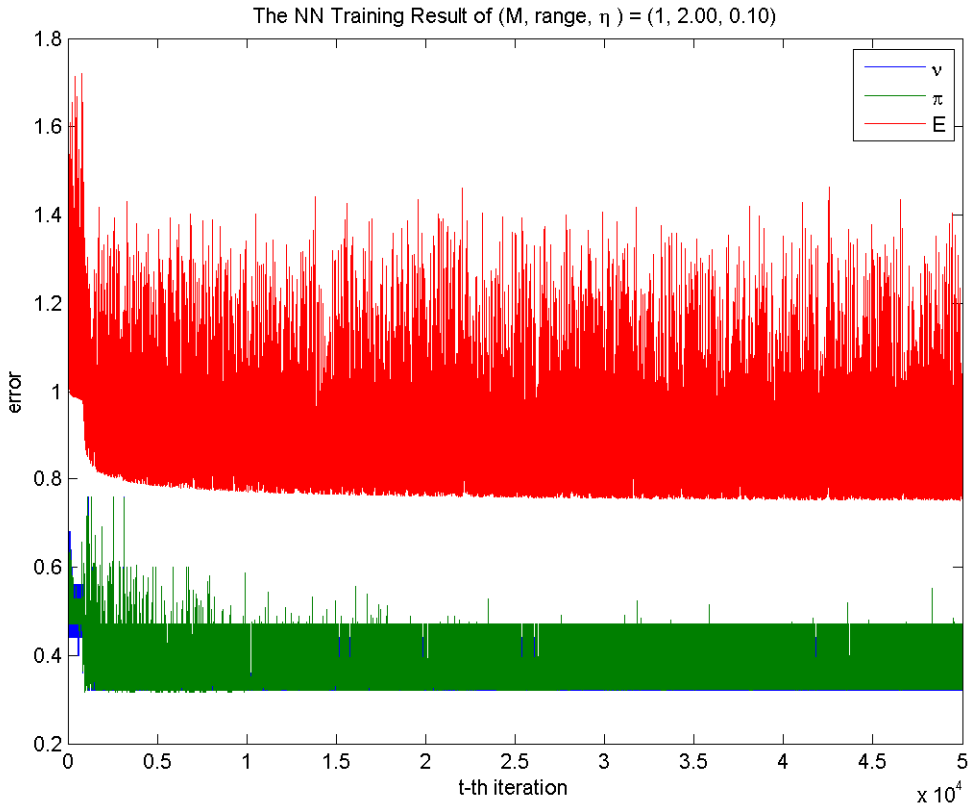

The NN Training Result of (M, range, η) = (1, 0.02, 1.00)
ν
π
E
error
t-th iteration
x 10^4

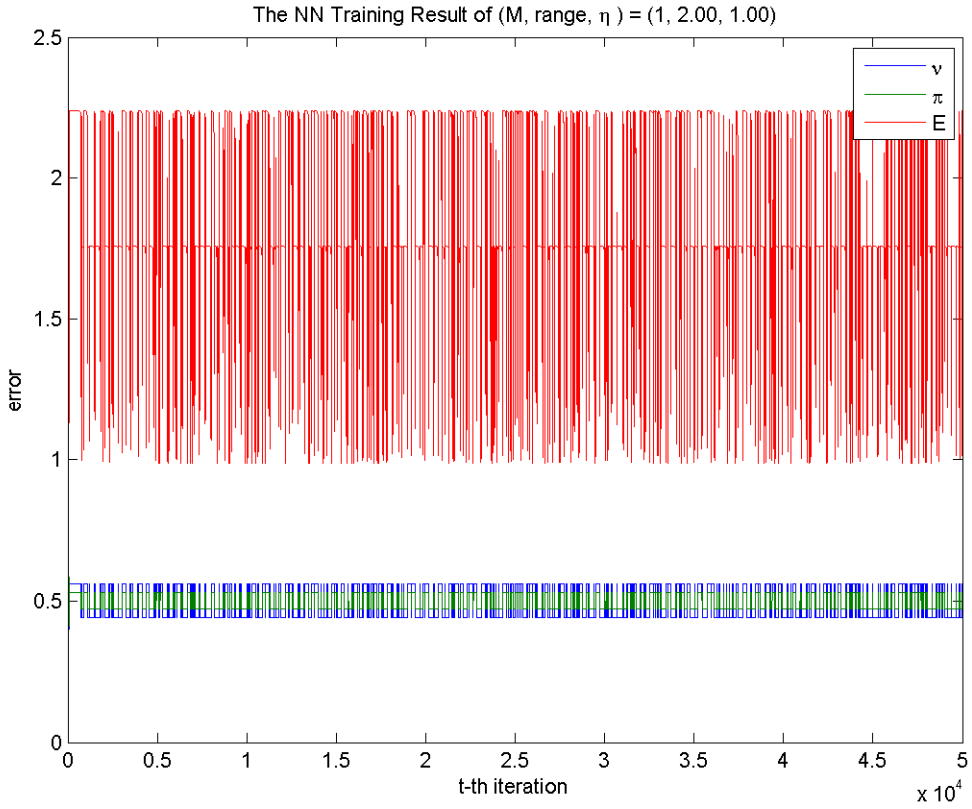

The NN Training Result of (M, range, η) = (1, 0.20, 0.01)
ν
π
E
error
t-th iteration
x 10^4

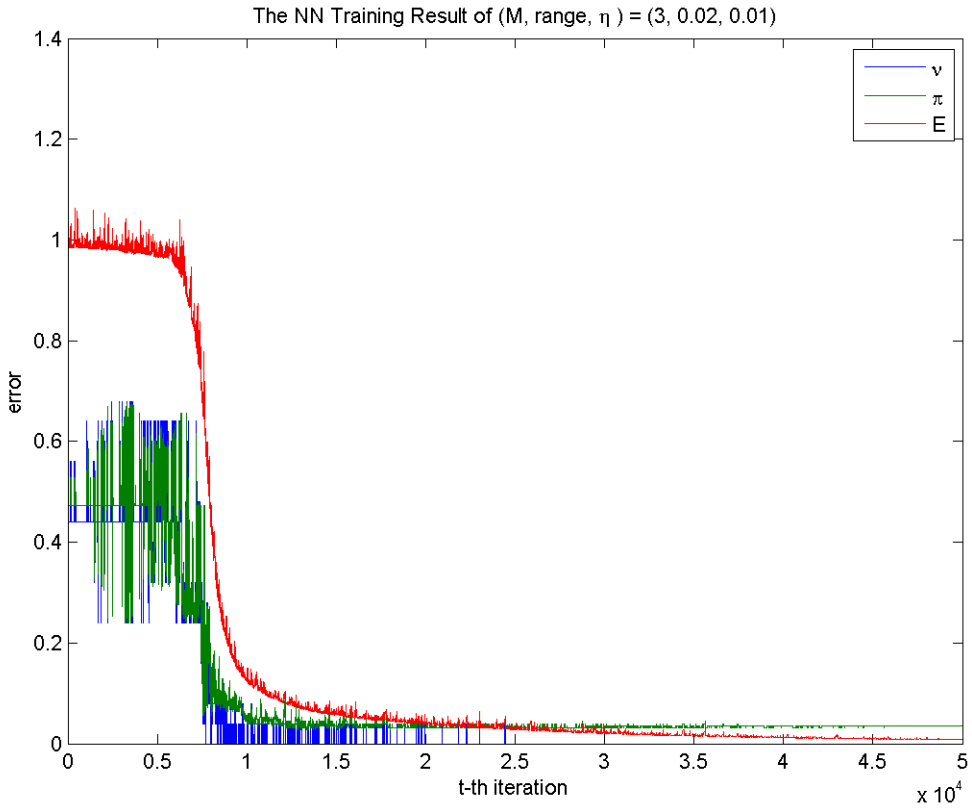

The NN Training Result of (M, range, η) = (1, 0.20, 0.10)
ν
π
E
error
t-th iteration
x 10^4

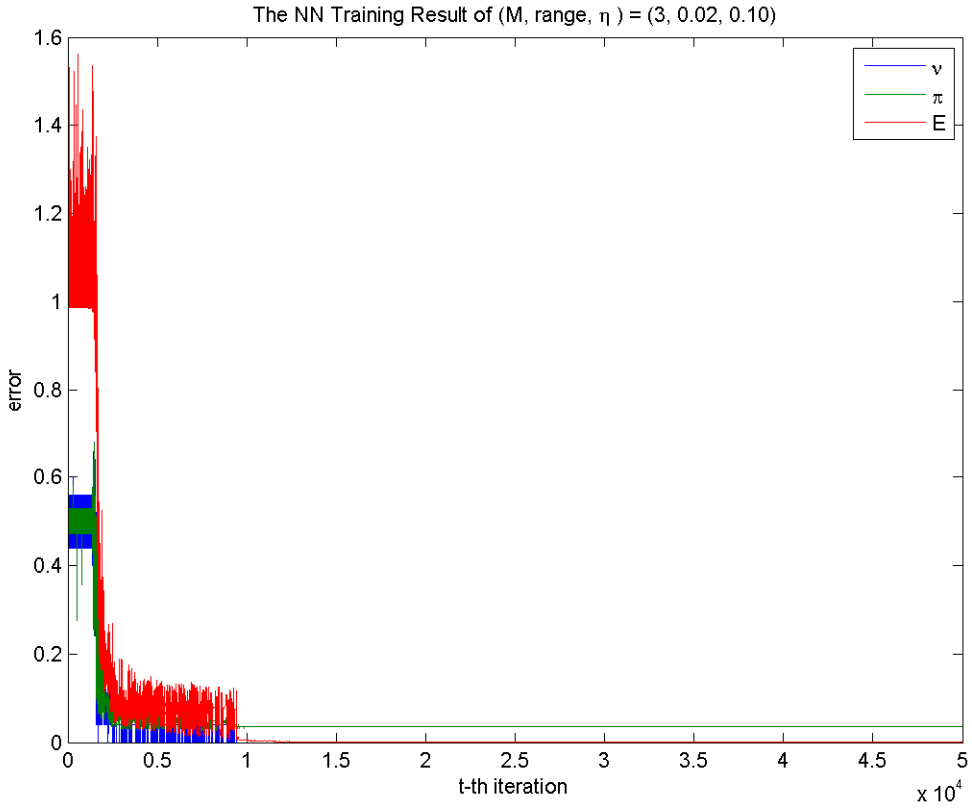

The NN Training Result of (M, range, η) = (1, 0.20, 1.00)
ν
π
E
error
t-th iteration
x 10^4

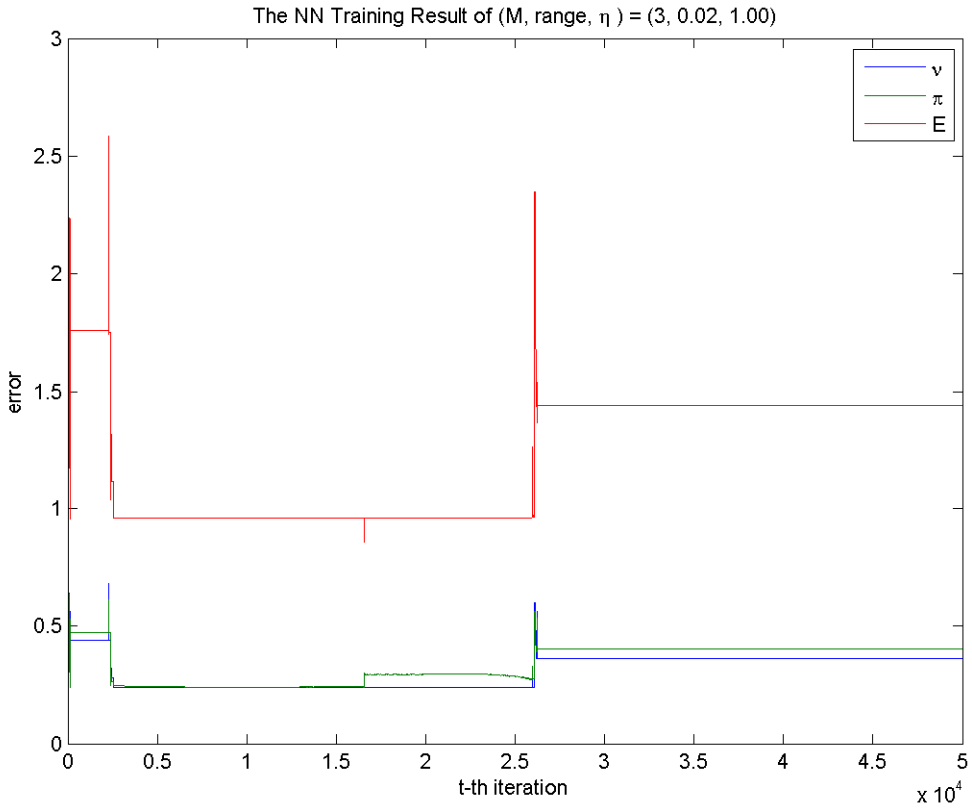

The NN Training Result of (M, range, η) = (1, 2.00, 0.01)
ν
π
E
error
t-th iteration
x 10^4

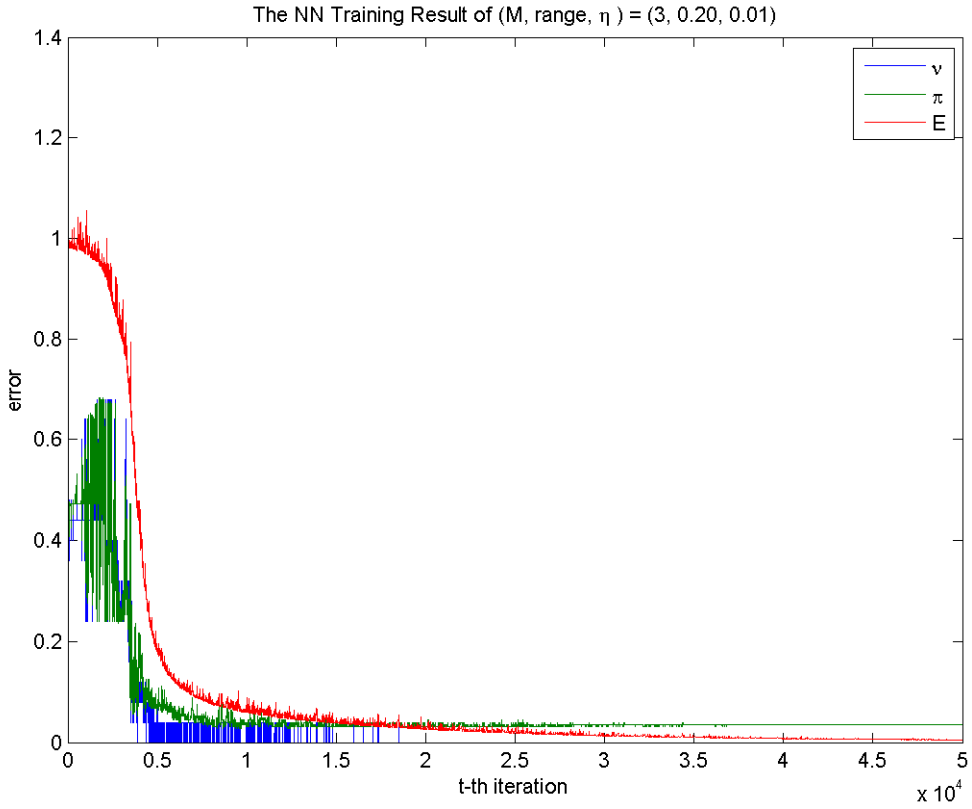

The NN Training Result of (M, range, η) = (1, 2.00, 0.10)
ν
π
E
error
t-th iteration
x 10^4

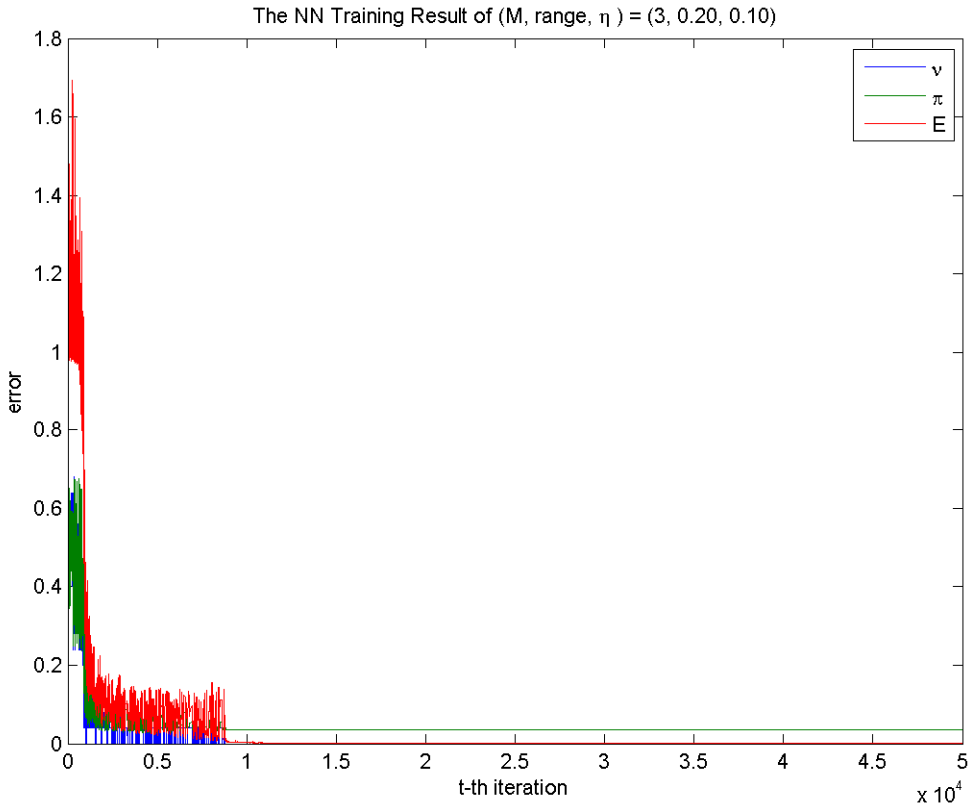

The NN Training Result of (M, range, η) = (1, 2.00, 1.00)
ν
π
E
error
t-th iteration
x 10^4

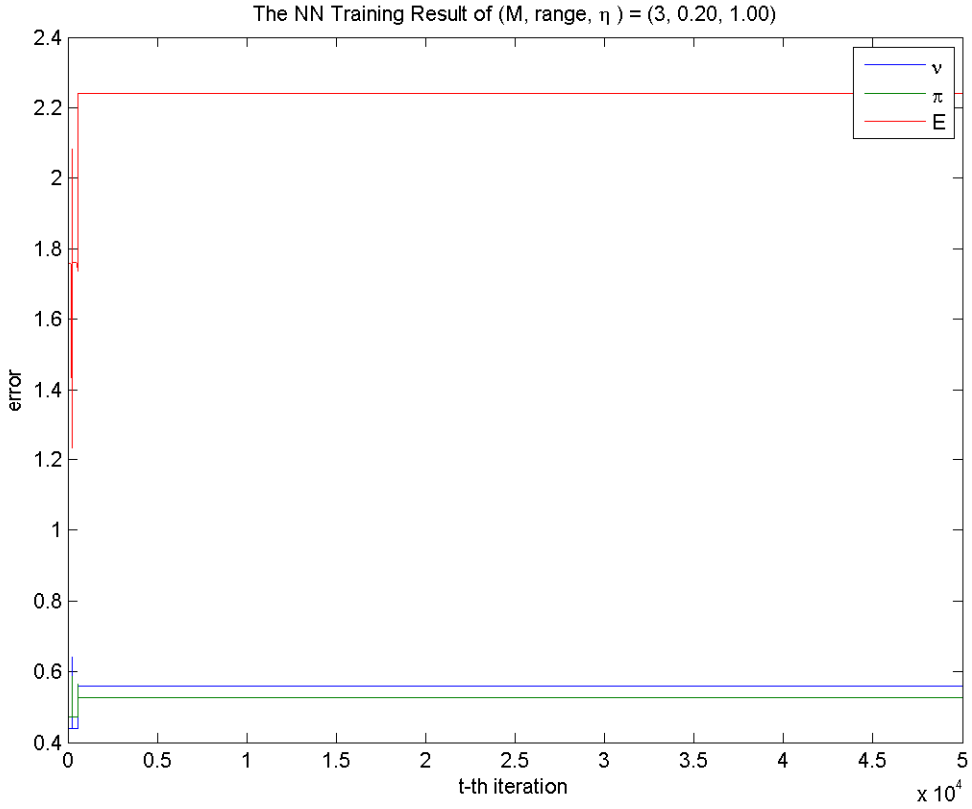

The NN Training Result of (M, range, η) = (3, 0.02, 0.01)
ν
π
E
error
t-th iteration
x 10^4

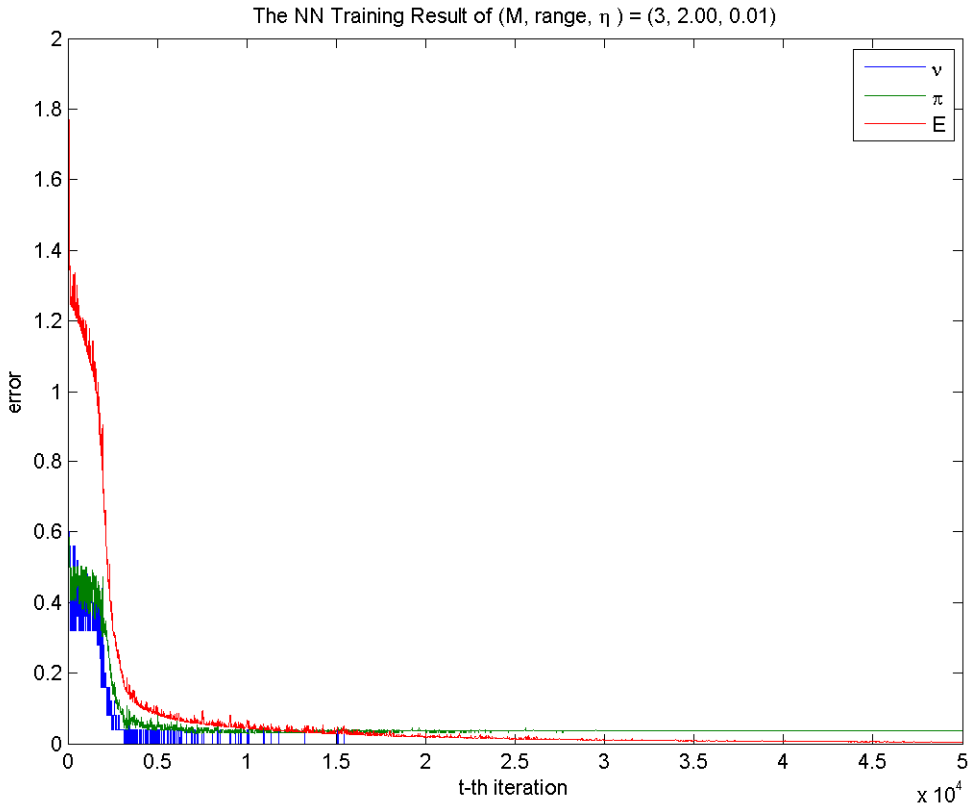

The NN Training Result of (M, range, η) = (3, 0.02, 0.10)
ν
π
E
error
t-th iteration
x 10^4

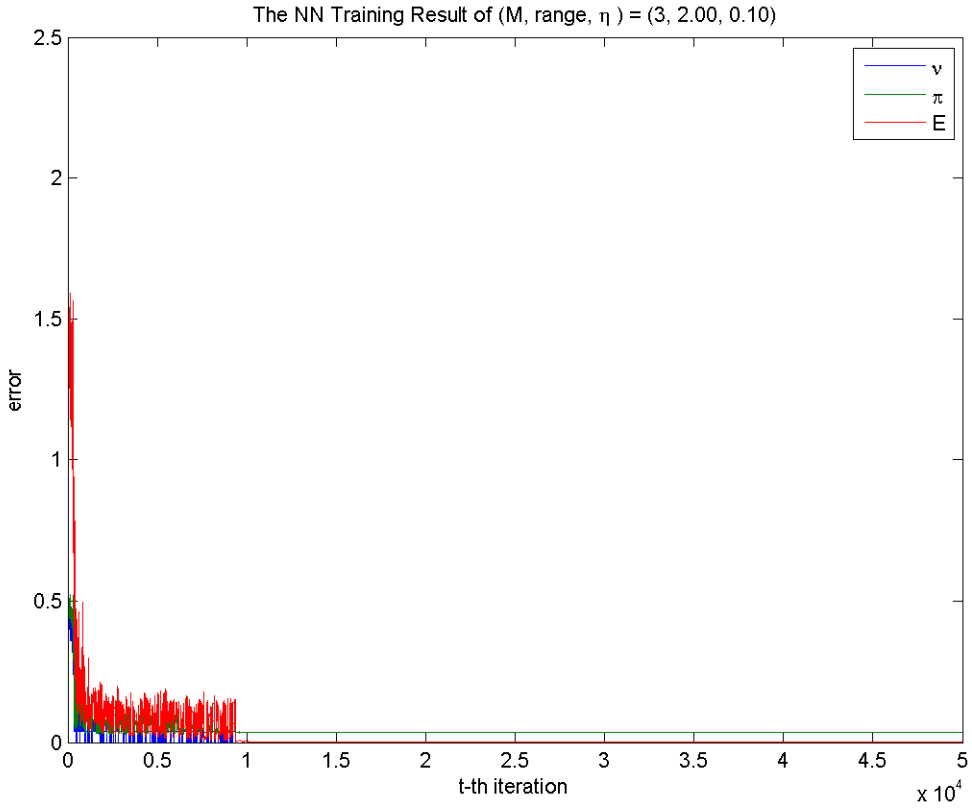

The NN Training Result of (M, range, η) = (3, 0.02, 1.00)
ν
π
E
error
t-th iteration
x 10^4

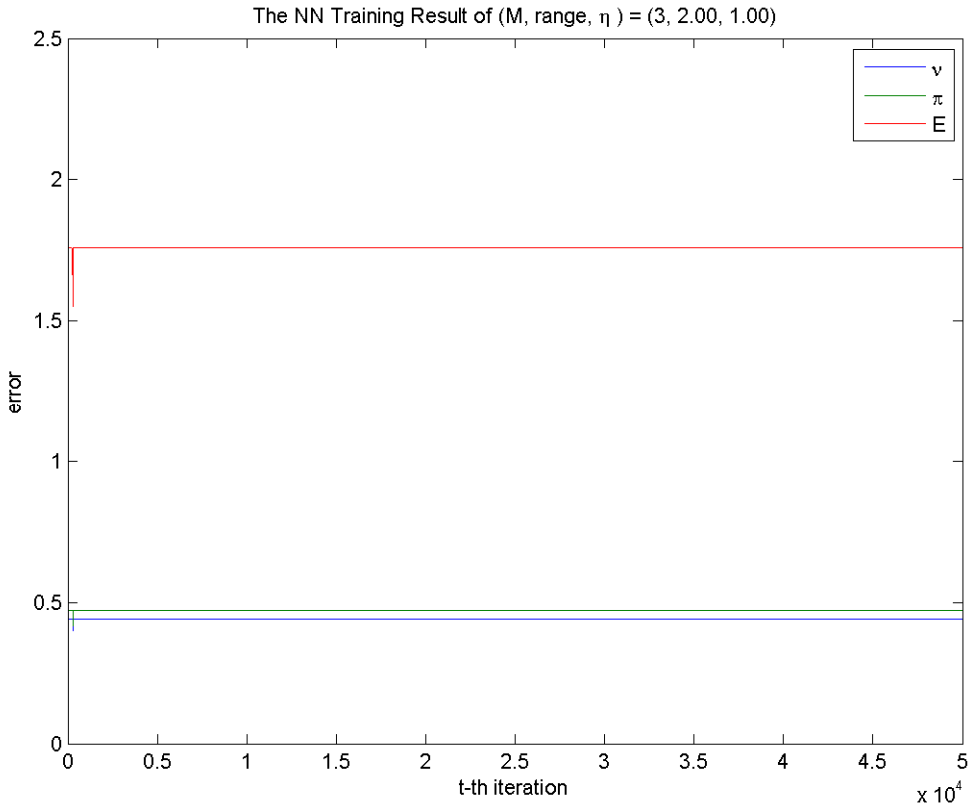

The NN Training Result of (M, range, η) = (3, 0.20, 0.01)
ν
π
E
error
t-th iteration
x 10^4

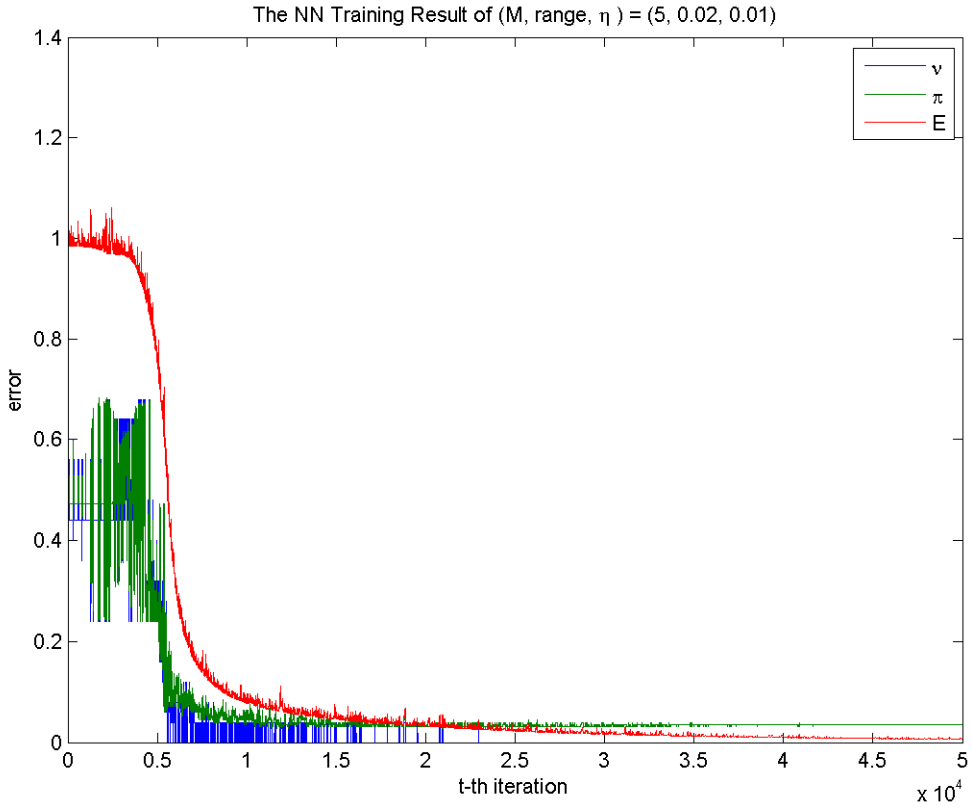

The NN Training Result of (M, range, η) = (3, 0.20, 0.10)
ν
π
E
error
t-th iteration
x 10^4

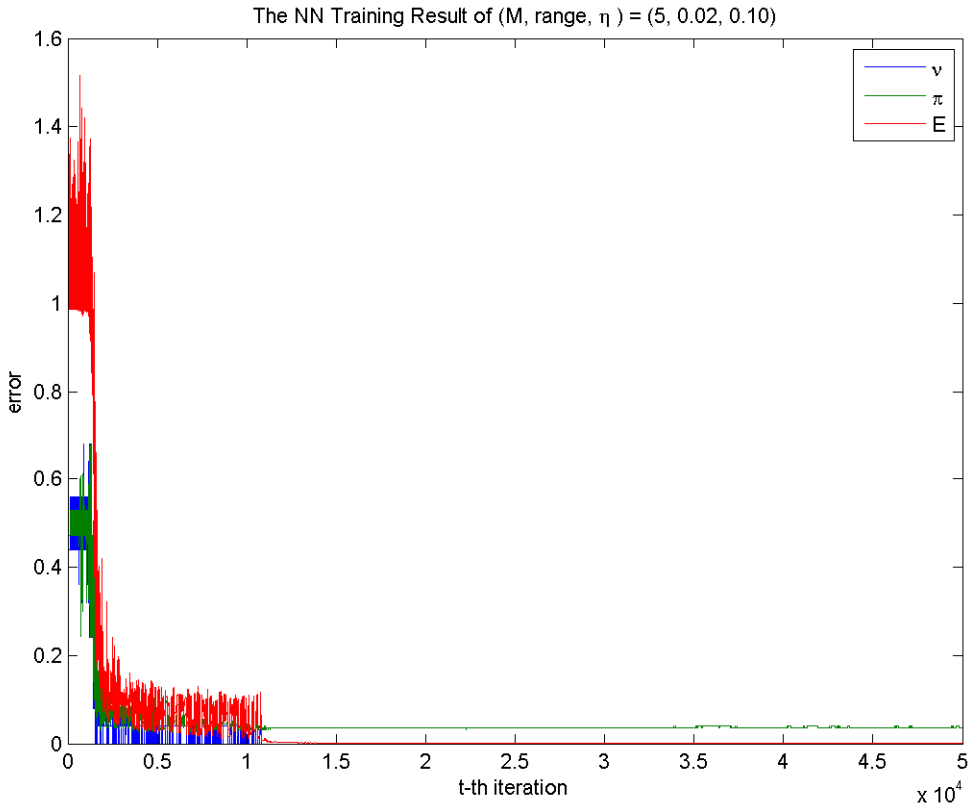

The NN Training Result of (M, range, η) = (3, 0.20, 1.00)
ν
π
E
error
t-th iteration
x 10^4

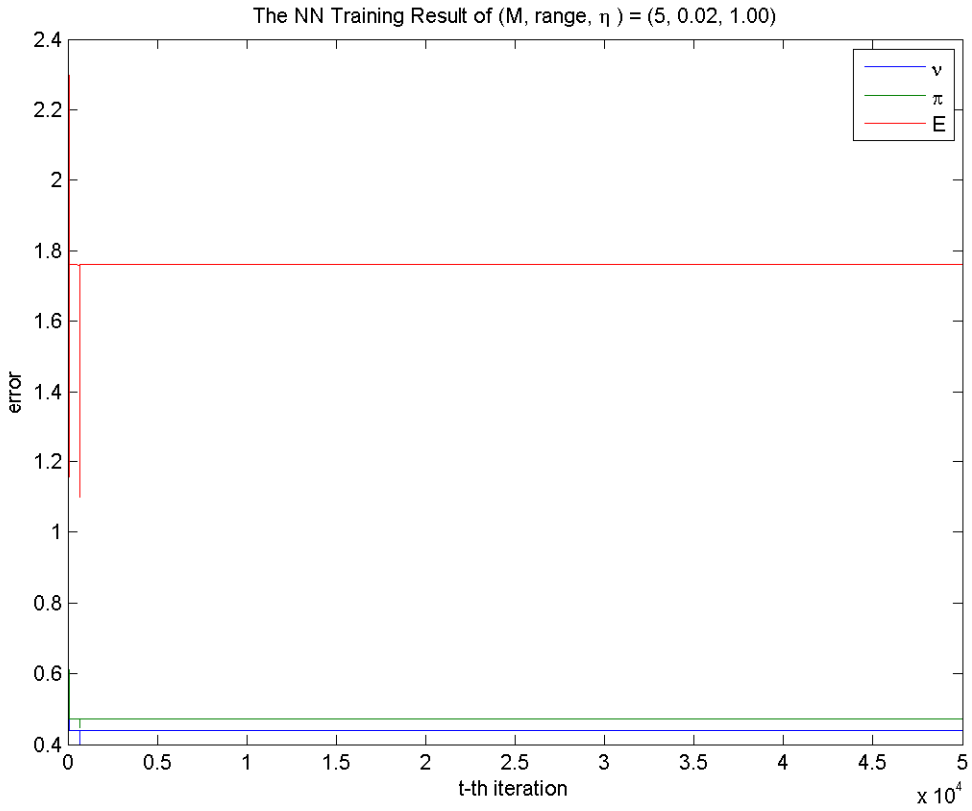

The NN Training Result of (M, range, η) = (3, 2.00, 0.01)
ν
π
E
error
t-th iteration
x 10^4

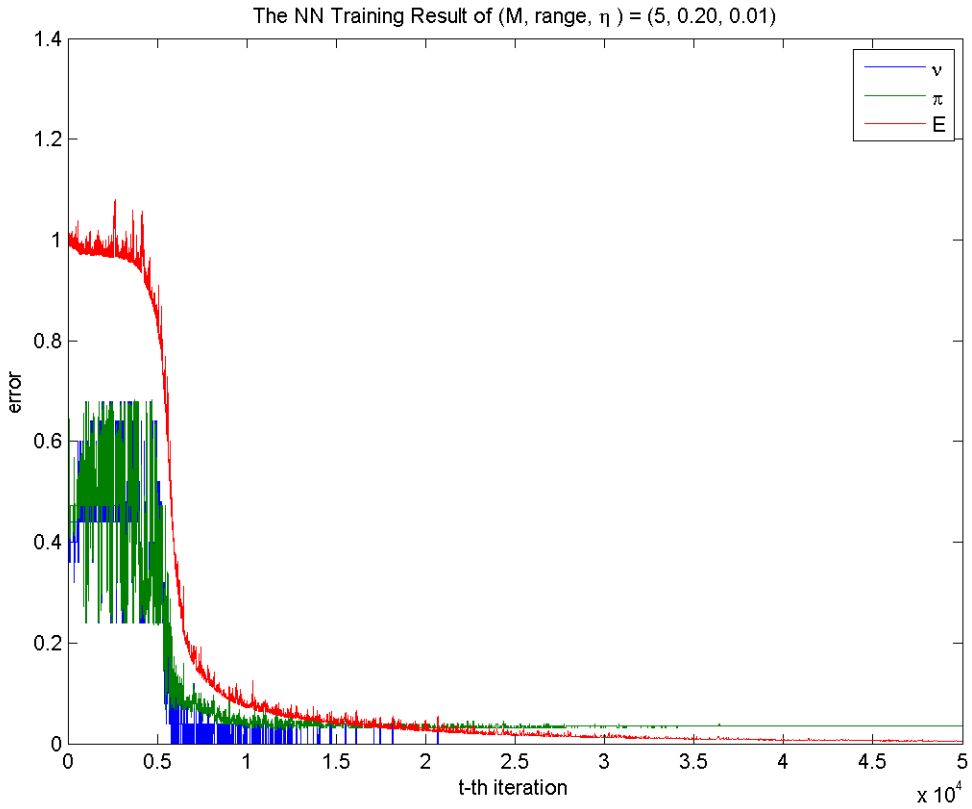

The NN Training Result of (M, range, η) = (3, 2.00, 0.10)
ν
π
E
error
t-th iteration
x 10^4

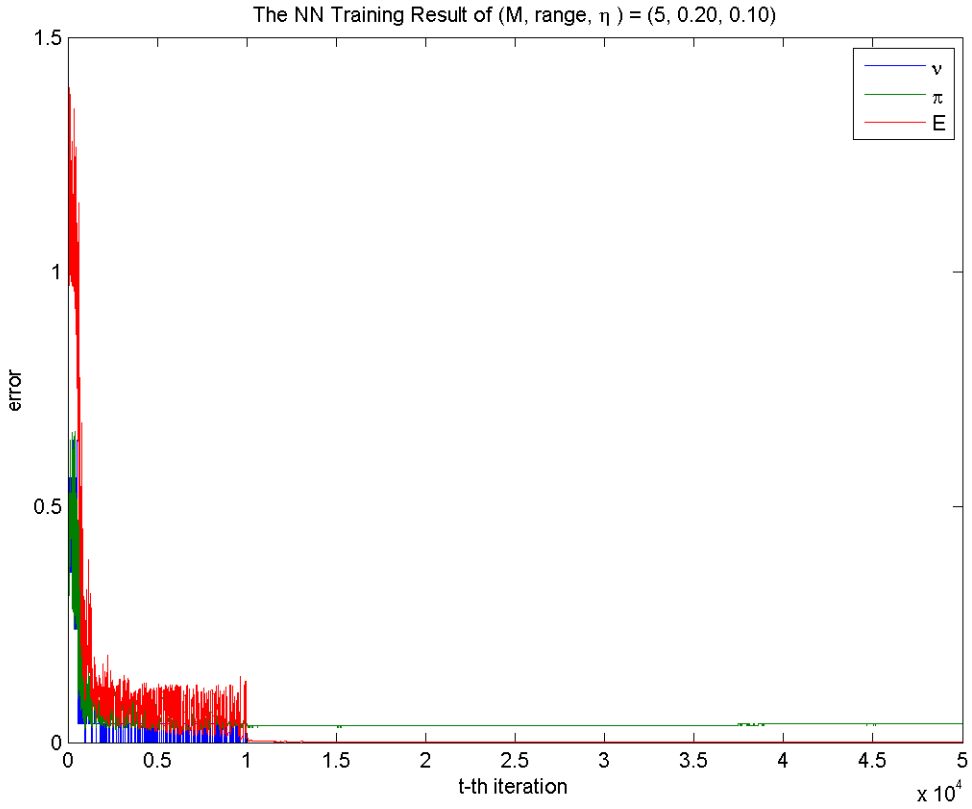

The NN Training Result of (M, range, η) = (3, 2.00, 1.00)
ν
π
E
error
t-th iteration
x 10^4

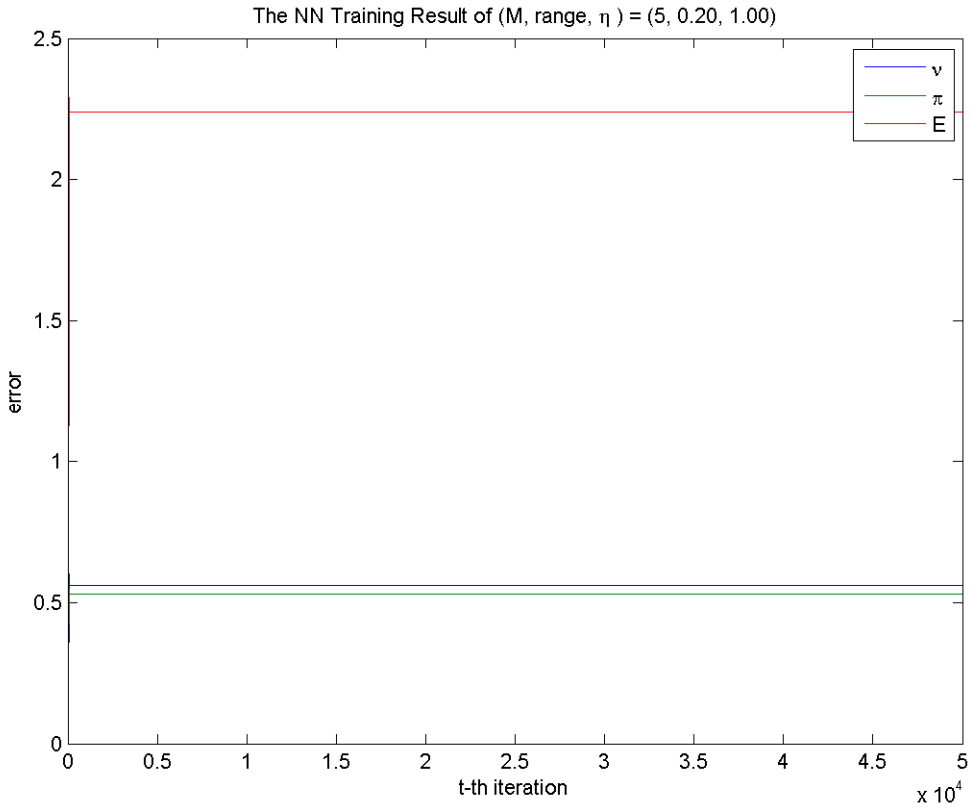

The NN Training Result of (M, range, η) = (5, 0.02, 0.01)
ν
π
E
error
t-th iteration
x 10^4

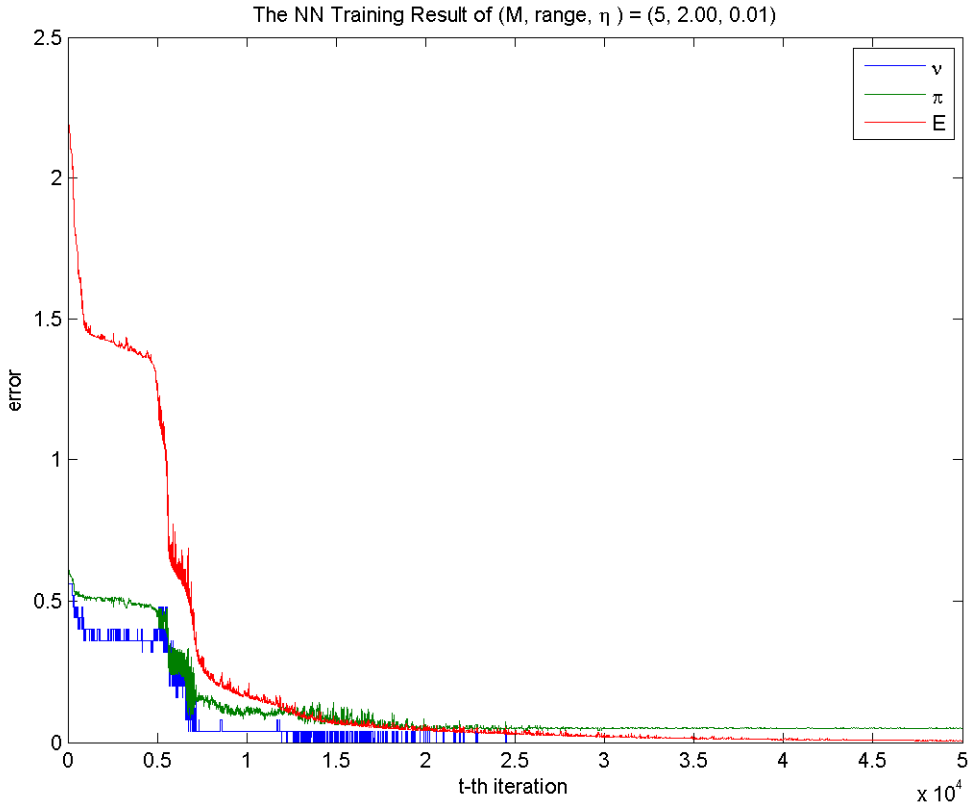

The NN Training Result of (M, range, η) = (5, 0.02, 0.10)
ν
π
E
error
t-th iteration
x 10^4

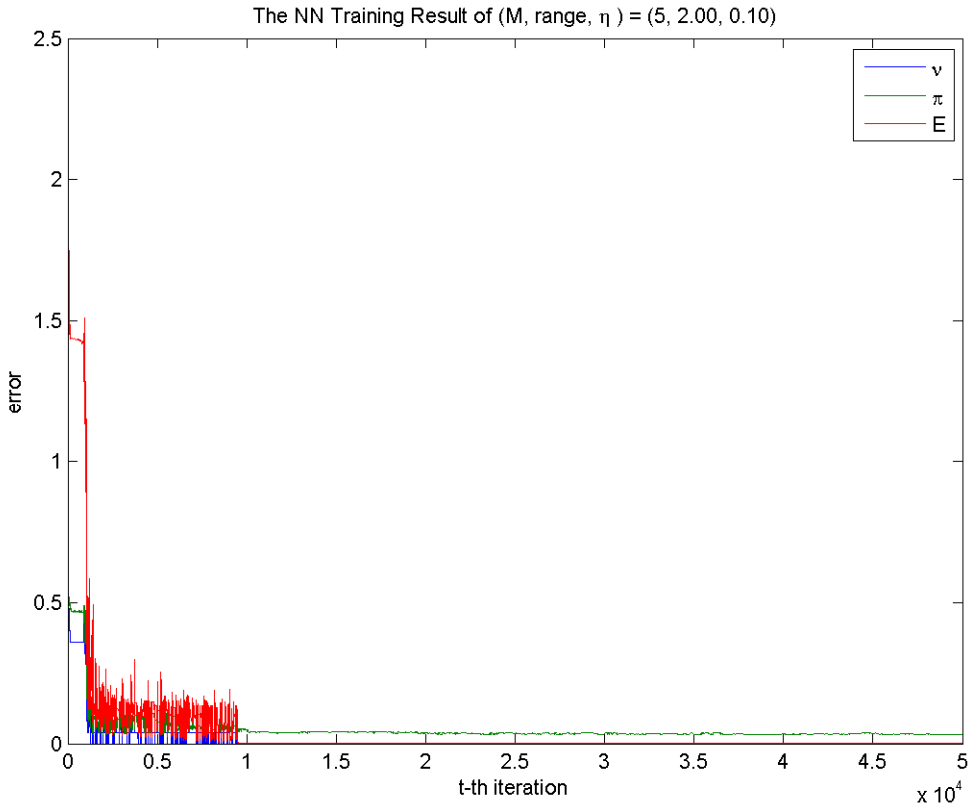

The NN Training Result of (M, range, η) = (5, 0.02, 1.00)
ν
π
E
error
t-th iteration
x 10^4

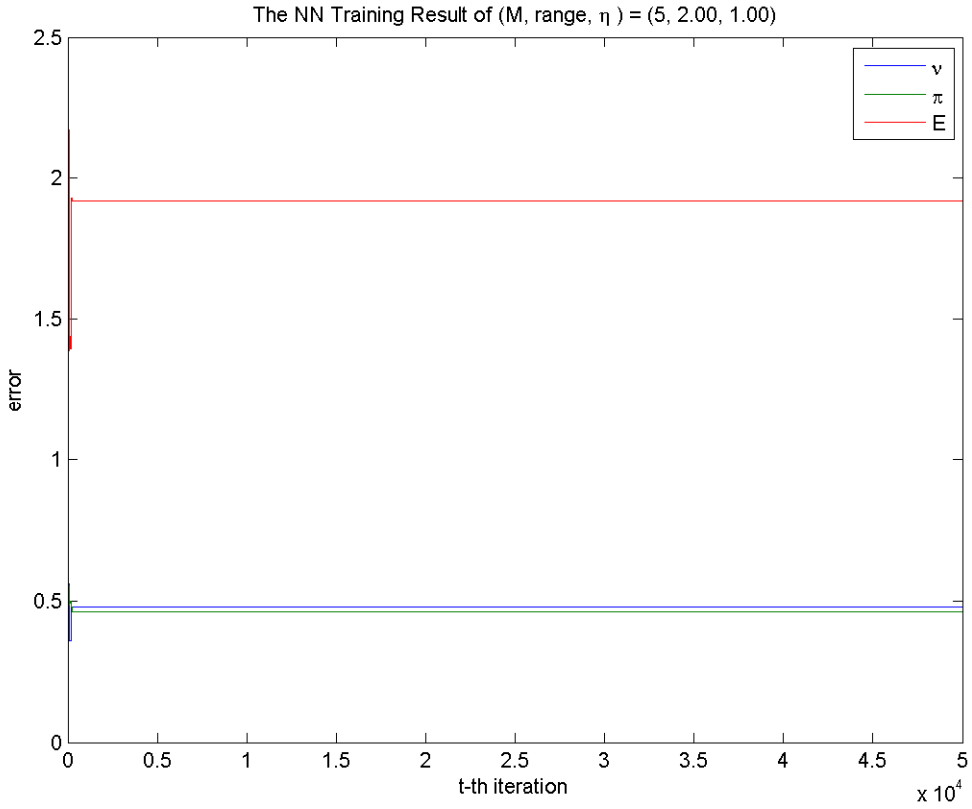

The NN Training Result of (M, range, η) = (5, 0.20, 0.01)
ν
π
E
error
t-th iteration
x 10^4


The NN Training Result of (M, range, η) = (5, 0.20, 0.10)
ν
π
E
error
t-th iteration
x 10^4


The NN Training Result of (M, range, η) = (5, 0.20, 1.00)
ν
π
E
error
t-th iteration
x 10^4


The NN Training Result of (M, range, η) = (5, 2.00, 0.01)
ν
π
E
error
t-th iteration
x 10^4


The NN Training Result of (M, range, η) = (5, 2.00, 0.10)
ν
π
E
error
t-th iteration
x 10^4


The NN Training Result of (M, range, η) = (5, 2.00, 1.00)
ν
π
E
error
t-th iteration
x 10^4