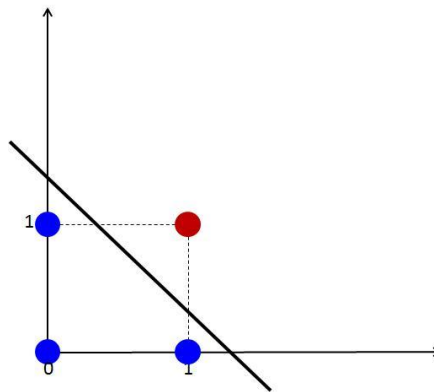


Corso Intelligenza Artificiale
Parte I: Reti Neurali Artificiali
Esercitazione per il Lab

**1. Problemi di separabilità lineare con modelli neuronali del tipo
*Perceptron***

1.1. Operatore congiunzione logica (AND)

1.1.1. Rappresentazione geometrica del problema



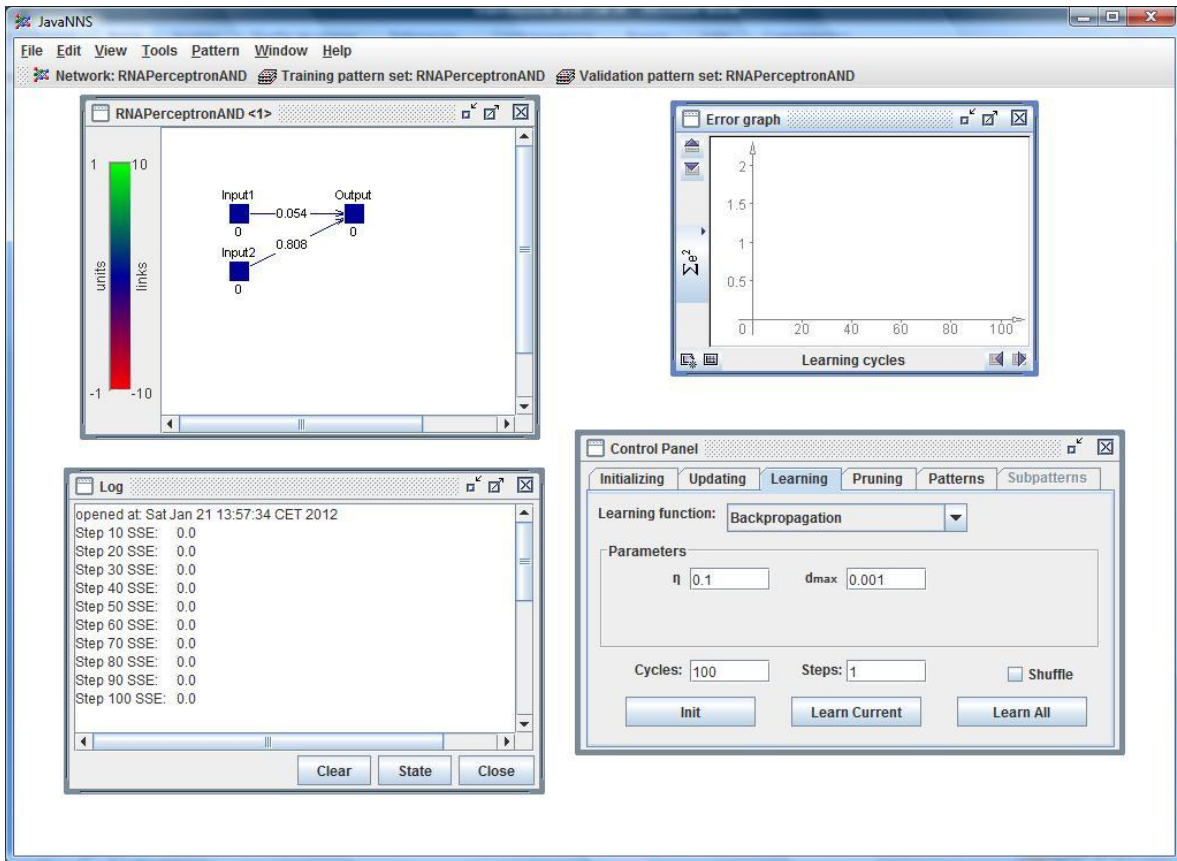
1.1.2. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 4
No. of input units : 2
No. of output units : 1

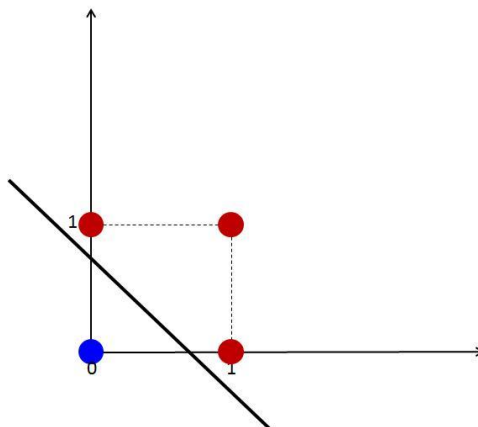
```
# Input pattern 1:
1 1
# Output pattern 1:
1
# Input pattern 2:
1 0
# Output pattern 2:
0
# Input pattern 3:
0 1
# Output pattern 3:
0
# Input pattern 4:
0 0
# Output pattern 4:
0
```

1.1.3. L'implementazione via JavaNNS



1.2. Operatore disgiunzione logica inclusiva (OR)

1.2.1. Rappresentazione geometrica del problema



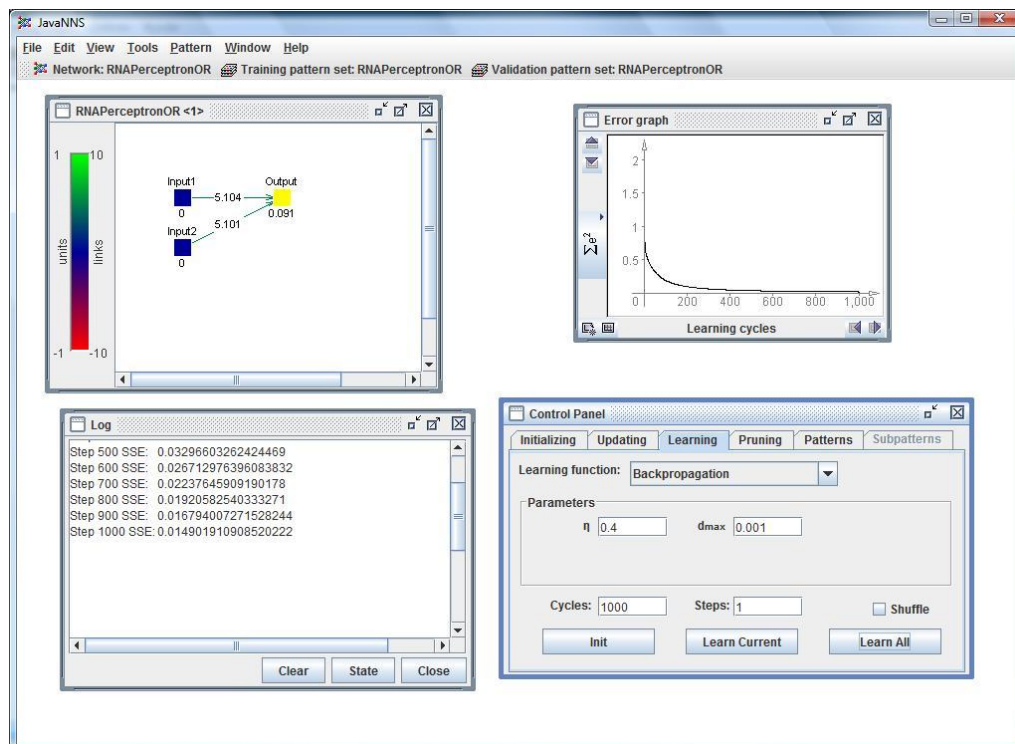
1.2.2. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 4
No. of input units : 2
No. of output units : 1

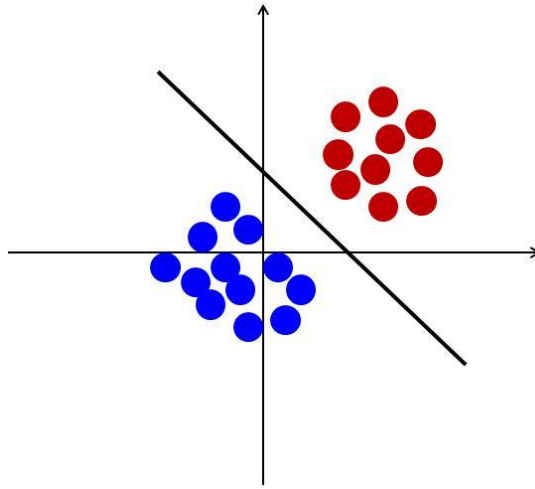
```
# Input pattern 1:  
1 1  
# Output pattern 1:  
1  
# Input pattern 2:  
1 0  
# Output pattern 2:  
1  
# Input pattern 3:  
0 1  
# Output pattern 3:  
1  
# Input pattern 4:  
0 0  
# Output pattern 4:  
0
```

1.2.3. L'implementazione via JavaNNS



1.3. Altri problemi di separabilità lineare

1.3.1. Rappresentazione geometrica del problema



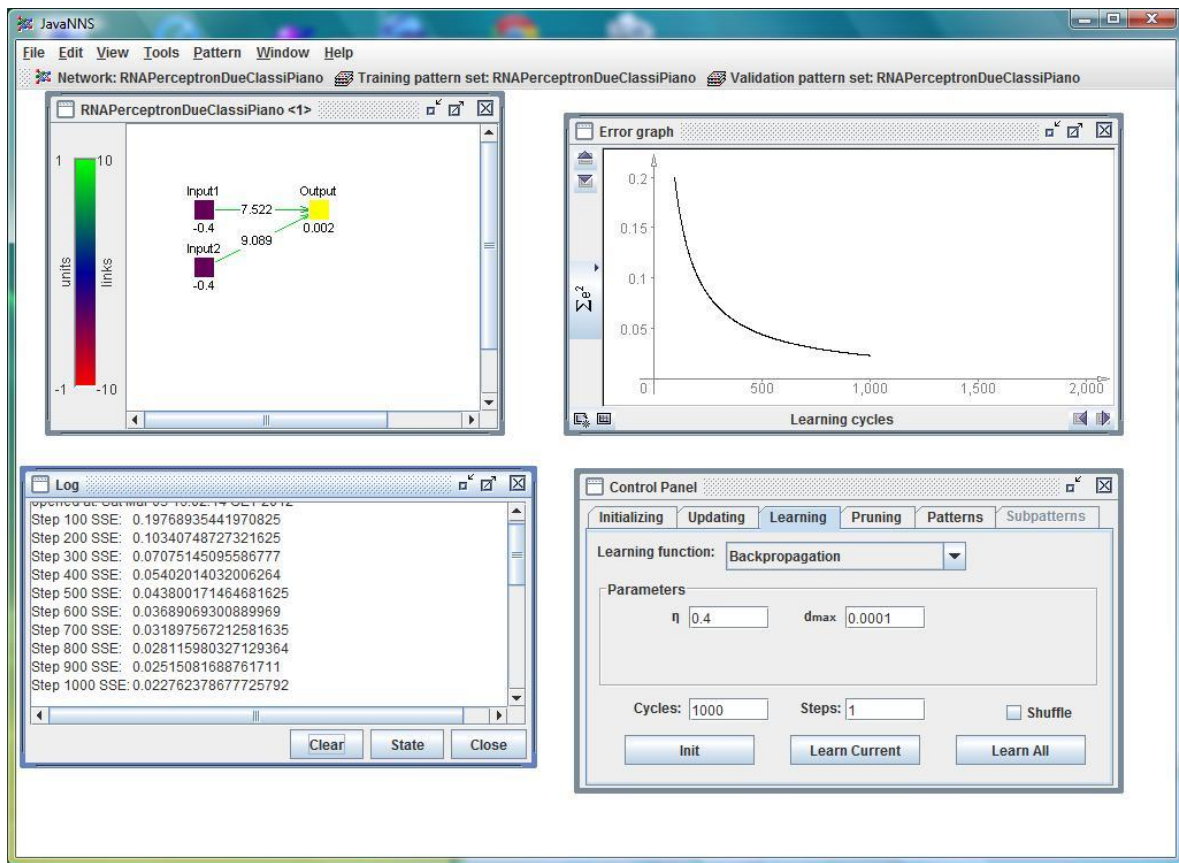
1.3.2. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 26
No. of input units : 2
No. of output units : 1

# Input pattern 1: 0.1 0.1 # Output pattern 1: 1 # Input pattern 2: 0.1 0.2 # Output pattern 2: 1 # Input pattern 3: 0.1 0.3 # Output pattern 3: 1 # Input pattern 4: 0.2 0.1 # Output pattern 4: 1 # Input pattern 5: 0.2 0.2 # Output pattern 5: 1 # Input pattern 6: 0.2 0.3 # Output pattern 6: 1 # Input pattern 7: 0.2 0.4 # Output pattern 7: 1 # Input pattern 8: 0.3 0.1 # Output pattern 8: 1 # Input pattern 9: 0.3 0.2 # Output pattern 9: 1	# Input pattern 10: 0.3 0.3 # Output pattern 10: 1 # Input pattern 11: 0.3 0.4 # Output pattern 11: 1 # Input pattern 12: 0.4 0.1 # Output pattern 12: 1 # Input pattern 13: 0.4 0.2 # Output pattern 13: 1 # Input pattern 14: 0.4 0.3 # Output pattern 14: 1 # Input pattern 15: -0.1 -0.3 # Output pattern 15: 0 # Input pattern 16: -0.1 -0.4 # Output pattern 16: 0 # Input pattern 17: -0.2 -0.2 # Output pattern 17: 0 # Input pattern 18: -0.2 -0.3 # Output pattern 18: 0 # Input pattern 19: -0.2 -0.4 # Output pattern 19: 0	# Input pattern 20: -0.2 -0.5 # Output pattern 20: 0 # Input pattern 21: -0.3 -0.2 # Output pattern 21: 0 # Input pattern 22: -0.3 -0.3 # Output pattern 22: 0 # Input pattern 23: -0.3 -0.4 # Output pattern 23: 0 # Input pattern 24: -0.3 -0.5 # Output pattern 24: 0 # Input pattern 25: -0.4 -0.3 # Output pattern 25: 0 # Input pattern 26: -0.4 -0.4 # Output pattern 26: 0
---	--	---

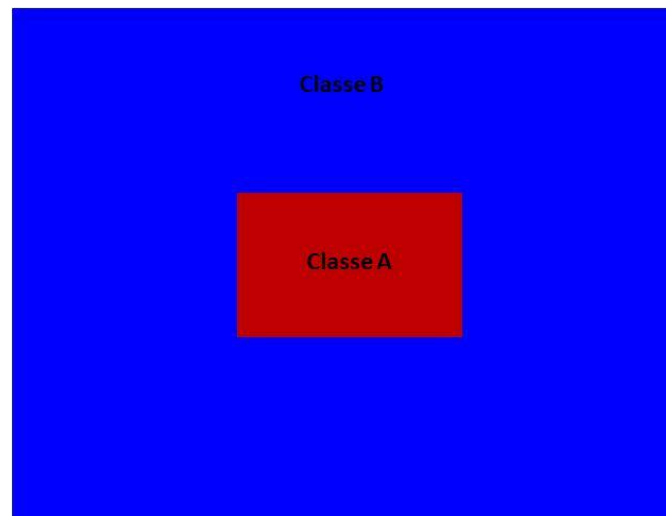
1.3.3. L'implementazione via JavaNNS



2. Reti neurali multistrato con l'algoritmo *Backpropagation*

2.1. Isola (due classi non linearmente separabili)

2.1.1. Rappresentazione geometrica del problema



2.1.2. I pattern d'ingresso

SNNS pattern definition file V3.2

generated at Mon Apr 25 15:58:23 1994

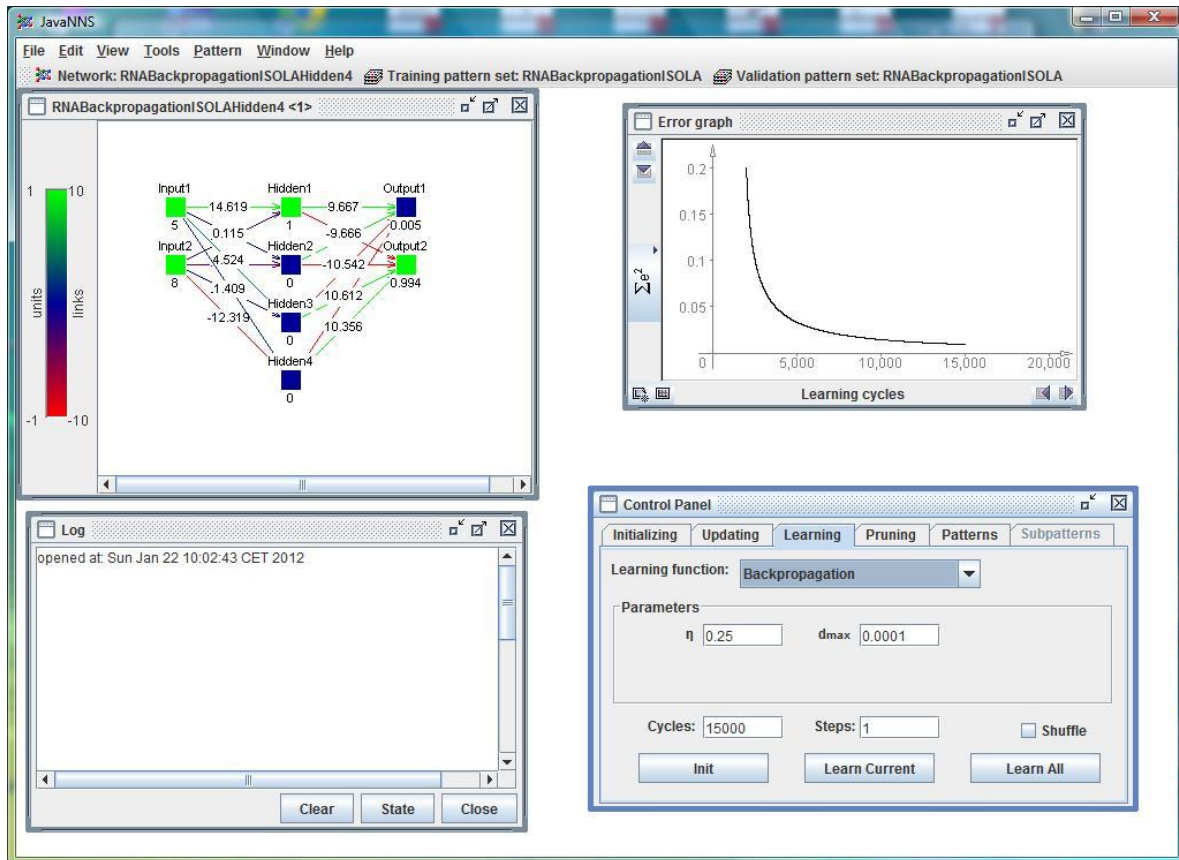
No. of patterns : 45

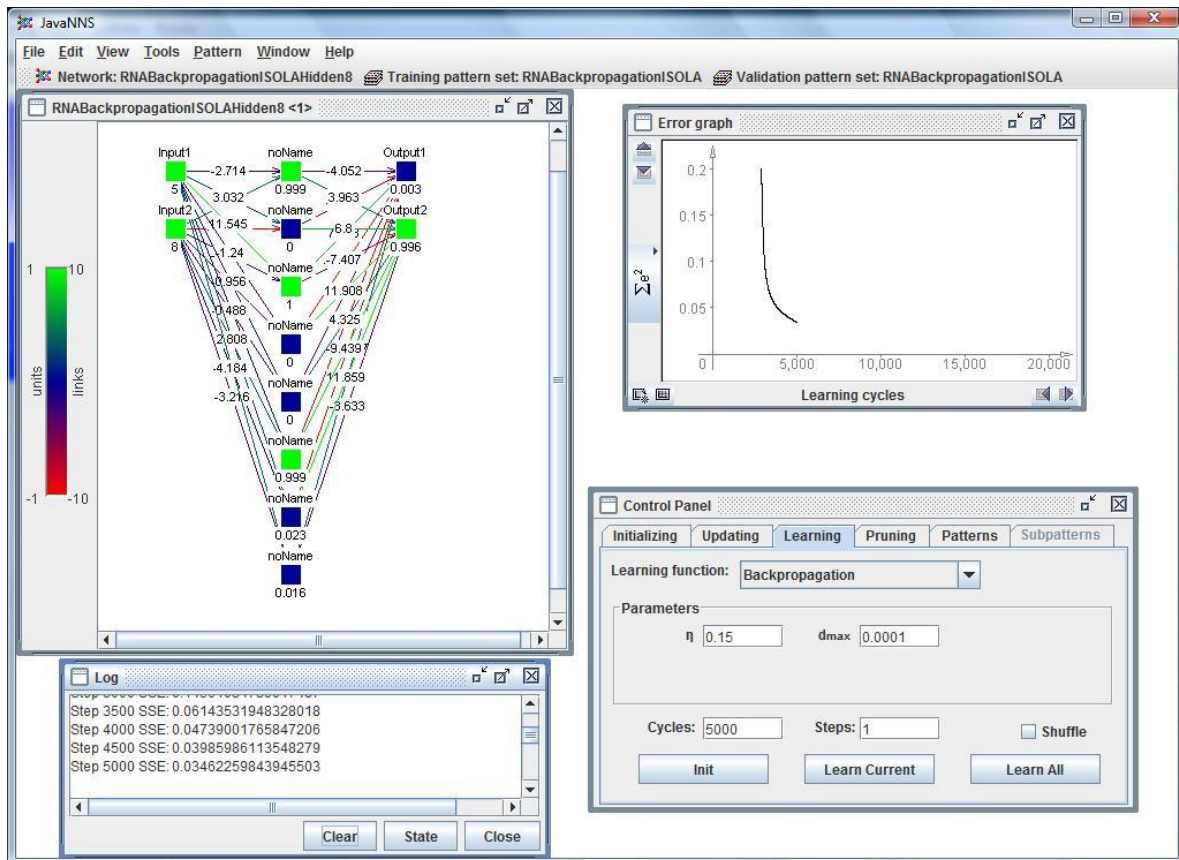
No. of input units : 2

No. of output units : 2

# Input pattern 1: 2 2 # Output pattern 1: 1 0 # Input pattern 2: 2 3 # Output pattern 2: 1 0 # Input pattern 3: 2 4 # Output pattern 3: 1 0 # Input pattern 4: 2 5 # Output pattern 4: 1 0 # Input pattern 5: 3 2 # Output pattern 5: 1 0 # Input pattern 6: 3 3 # Output pattern 6: 1 0 # Input pattern 7: 3 4 # Output pattern 7: 1 0 # Input pattern 8: 3 5 # Output pattern 8: 1 0 # Input pattern 9: 4 2 # Output pattern 9: 1 0 # Input pattern 10: 4 3 # Output pattern 10: 1 0	# Input pattern 11: 4 4 # Output pattern 11: 1 0 # Input pattern 12: 4 5 # Output pattern 12: 1 0 # Input pattern 13: 5 2 # Output pattern 13: 1 0 # Input pattern 14: 5 3 # Output pattern 14: 1 0 # Input pattern 15: 5 4 # Output pattern 15: 1 0 # Input pattern 16: 5 5 # Output pattern 16: 1 0 # Input pattern 17: 6 2 # Output pattern 17: 1 0 # Input pattern 18: 6 3 # Output pattern 18: 1 0 # Input pattern 19: 6 4 # Output pattern 19: 1 0 # Input pattern 20: 6 5 # Output pattern 20: 1 0	# Input pattern 21: 1 1 # Output pattern 21: 0 1 # Input pattern 22: 1 2 # Output pattern 22: 0 1 # Input pattern 23: 1 3 # Output pattern 23: 0 1 # Input pattern 24: 1 4 # Output pattern 24: 0 1 # Input pattern 25: 1 5 # Output pattern 25: 0 1 # Input pattern 26: 1 6 # Output pattern 26: 0 1 # Input pattern 27: 2 1 # Output pattern 27: 0 1 # Input pattern 28: 3 1 # Output pattern 28: 0 1 # Input pattern 29: 4 1 # Output pattern 29: 0 1 # Input pattern 30: 5 1 # Output pattern 30: 0 1	# Input pattern 31: 6 1 # Output pattern 31: 0 1 # Input pattern 32: 7 1 # Output pattern 32: 0 1 # Input pattern 33: 7 2 # Output pattern 33: 0 1 # Input pattern 34: 7 3 # Output pattern 34: 0 1 # Input pattern 35: 7 5 # Output pattern 35: 0 1 # Input pattern 36: 7 7 # Output pattern 36: 0 1 # Input pattern 37: 2 6 # Output pattern 37: 0 1 # Input pattern 38: 3 6 # Output pattern 38: 0 1 # Input pattern 39: 4 6 # Output pattern 39: 0 1 # Input pattern 40: 6 6 # Output pattern 40: 0 1	# Output pattern 41: 0 1 # Input pattern 42: 2 7 # Output pattern 42: 0 1 # Input pattern 43: 3 8 # Output pattern 43: 0 1 # Input pattern 44: 4 7 # Output pattern 44: 0 1 # Input pattern 45: 5 8 # Output pattern 45: 0 1
--	--	--	--	---

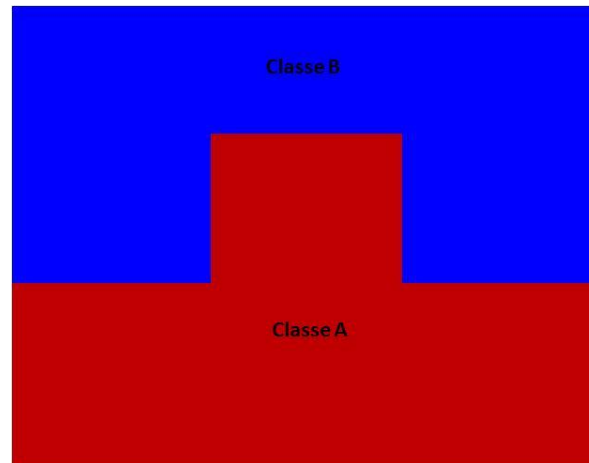
2.1.3. L'implementazione via JavaNNS





2.2. Penisola (due classi non linearmente separabili)

2.2.1. Rappresentazione geometrica del problema



2.2.2. I pattern d'ingresso

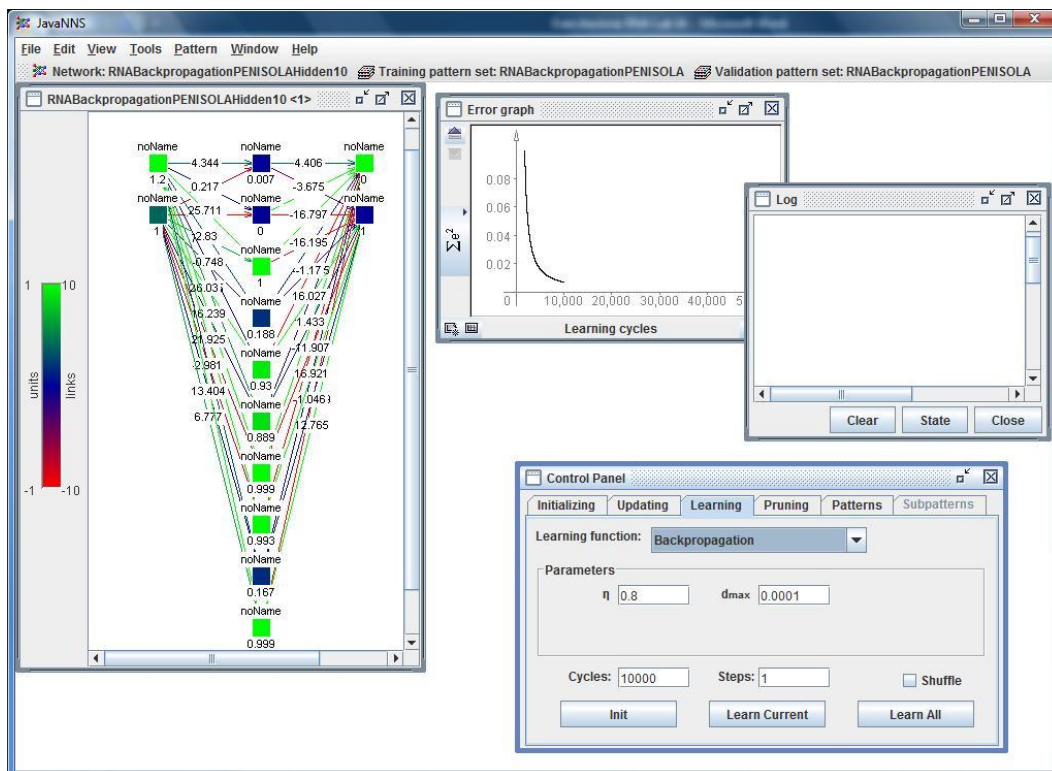
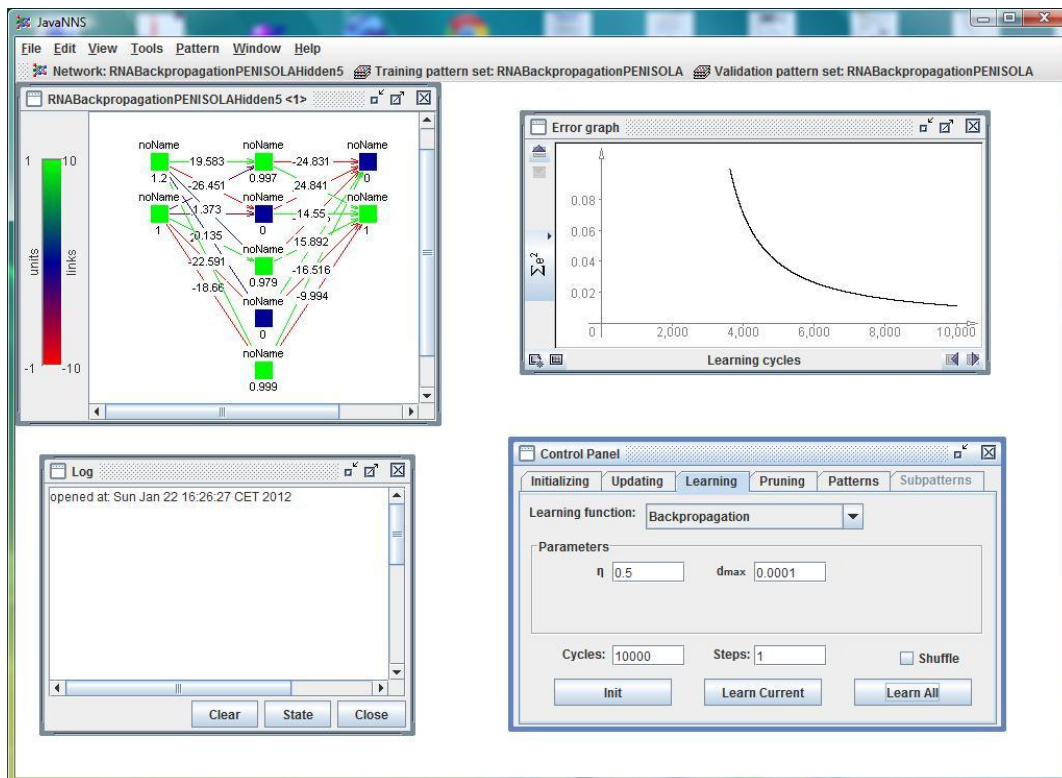
SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 76
No. of input units : 2
No. of output units : 2

# Input pattern 1: 0.1 0.1 # Output pattern 1: 1 0 # Input pattern 2: 0.1 0.2 # Output pattern 2: 1 0 # Input pattern 3: 0.1 0.4 # Output pattern 3: 1 0 # Input pattern 4: 0.2 0.1 # Output pattern 4: 1 0 # Input pattern 5: 0.2 0.2 # Output pattern 5: 1 0 # Input pattern 6: 0.2 0.4 # Output pattern 6: 1 0 # Input pattern 7: 0.3 0.1 # Output pattern 7: 1 0 # Input pattern 8: 0.3 0.2 # Output pattern 8: 1 0 # Input pattern 9: 0.3 0.4 # Output pattern 9: 1 0 # Input pattern 10: 0.4 0.1 # Output pattern 10: 1 0	# Input pattern 11: 0.4 0.2 # Output pattern 11: 1 0 # Input pattern 12: 0.4 0.4 # Output pattern 12: 1 0 # Input pattern 13: 0.4 0.5 # Output pattern 13: 1 0 # Input pattern 14: 0.4 0.6 # Output pattern 14: 1 0 # Input pattern 15: 0.4 0.8 # Output pattern 15: 1 0 # Input pattern 16: 0.5 0.1 # Output pattern 16: 1 0 # Input pattern 17: 0.5 0.2 # Output pattern 17: 1 0 # Input pattern 18: 0.5 0.4 # Output pattern 18: 1 0 # Input pattern 19: 0.5 0.6 # Output pattern 19: 1 0 # Input pattern 20: 0.5 0.8 # Output pattern 20: 1 0	# Input pattern 21: 0.7 0.1 # Output pattern 21: 1 0 # Input pattern 22: 0.7 0.2 # Output pattern 22: 1 0 # Input pattern 23: 0.7 0.4 # Output pattern 23: 1 0 # Input pattern 24: 0.7 0.6 # Output pattern 24: 1 0 # Input pattern 25: 0.7 0.8 # Output pattern 25: 1 0 # Input pattern 26: 0.8 0.1 # Output pattern 26: 1 0 # Input pattern 27: 0.8 0.2 # Output pattern 27: 1 0 # Input pattern 28: 0.8 0.3 # Output pattern 28: 1 0 # Input pattern 29: 0.8 0.5 # Output pattern 29: 1 0 # Input pattern 30: 0.8 0.7 # Output pattern 30: 1 0	# Input pattern 31: 0.8 0.8 # Output pattern 31: 1 0 # Input pattern 32: 0.9 0.2 # Output pattern 32: 1 0 # Input pattern 33: 0.9 0.4 # Output pattern 33: 1 0 # Input pattern 34: 1.0 0.1 # Output pattern 34: 1 0 # Input pattern 35: 1.0 0.3 # Output pattern 35: 1 0 # Input pattern 36: 1.0 0.4 # Output pattern 36: 1 0 # Input pattern 37: 1.2 0.1 # Output pattern 37: 1 0 # Input pattern 38: 1.2 0.2 # Output pattern 38: 1 0 # Input pattern 39: 1.2 0.4 # Output pattern 39: 1 0 # Input pattern 40: 0.1 0.5 # Output pattern 40: 0 1	# Input pattern 41: 0.2 0.5 # Output pattern 41: 0 1 # Input pattern 42: 0.3 0.5 # Output pattern 42: 0 1 # Input pattern 43: 0.1 0.6 # Output pattern 43: 0 1 # Input pattern 44: 0.2 0.6 # Output pattern 44: 0 1 # Input pattern 45: 0.3 0.6 # Output pattern 45: 0 1 # Input pattern 46: 0.1 0.8 # Output pattern 46: 0 1 # Input pattern 47: 0.2 0.8 # Output pattern 47: 0 1 # Input pattern 48: 0.3 0.8 # Output pattern 48:
--	--	--	--	---

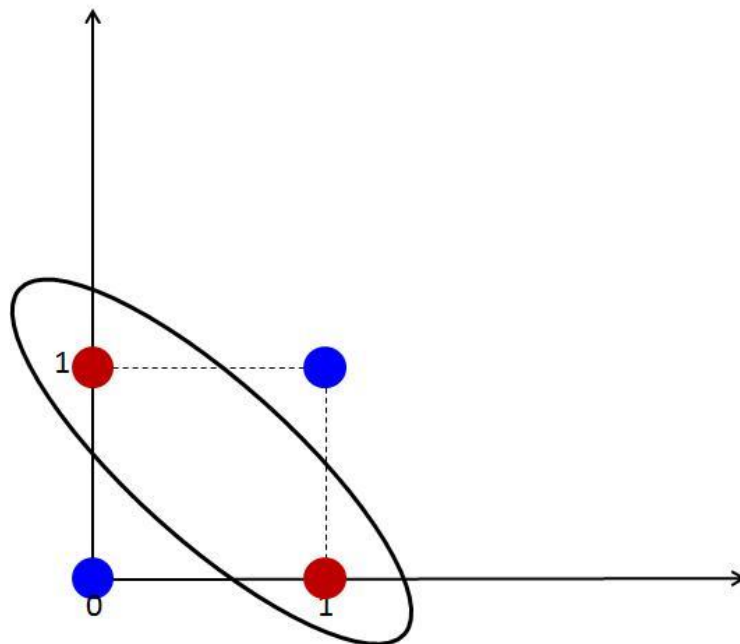
# Input pattern 49: 0.9 0.5 # Output pattern 49: 0 1 # Input pattern 50: 1.0 0.5 # Output pattern 50: 0 1 # Input pattern 51: 1.1 0.5 # Output pattern 51: 0 1 # Input pattern 52: 1.2 0.5 # Output pattern 52: 0 1 # Input pattern 53: 0.9 0.6 # Output pattern 53: 0 1 # Input pattern 54: 1.0 0.6 # Output pattern 54: 0 1 # Input pattern 55: 1.2 0.6 # Output pattern 55: 0 1 # Input pattern 54: 0.9 0.8 # Output pattern 56: 0 1 # Input pattern 57: 1.0 0.8 # Output pattern 57: 0 1	# Input pattern 58: 1.1 0.8 # Output pattern 58: 0 1 # Input pattern 59: 0.1 0.9 # Output pattern 59: 0 1 # Input pattern 60: 0.2 0.9 # Output pattern 60: 0 1 # Input pattern 61: 0.3 0.9 # Output pattern 61: 0 1 # Input pattern 62: 0.4 0.9 # Output pattern 62: 0 1 # Input pattern 63: 0.5 0.9 # Output pattern 63: 0 1 # Input pattern 64: 0.7 0.9 # Output pattern 64: 0 1 # Input pattern 65: 0.8 0.9 # Output pattern 65: 0 1 # Input pattern 66: 1.0 0.9 # Output pattern 66: 0 1	# Output pattern 67: 0 1 # Input pattern 68: 0.1 1.0 # Output pattern 68: 0 1 # Input pattern 69: 0.3 1.0 # Output pattern 69: 0 1 # Input pattern 70: 0.4 1.0 # Output pattern 70: 0 1 # Input pattern 71: 0.5 1.0 # Output pattern 71: 0 1 # Input pattern 72: 0.7 1.0 # Output pattern 72: 0 1 # Input pattern 73: 0.8 1.0 # Output pattern 73: 0 1 # Input pattern 74: 0.9 1.0 # Output pattern 74: 0 1 # Input pattern 75: 1.1 1.0 # Output pattern 75: 0 1 # Input pattern 76: 1.2 1.0 # Output pattern 76: 0 1		
---	---	--	--	--

2.2.3. L'implementazione via JavaNNS



2.3. Operatore disgiunzione logica esclusiva

2.3.1. Rappresentazione geometrica



2.3.2. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

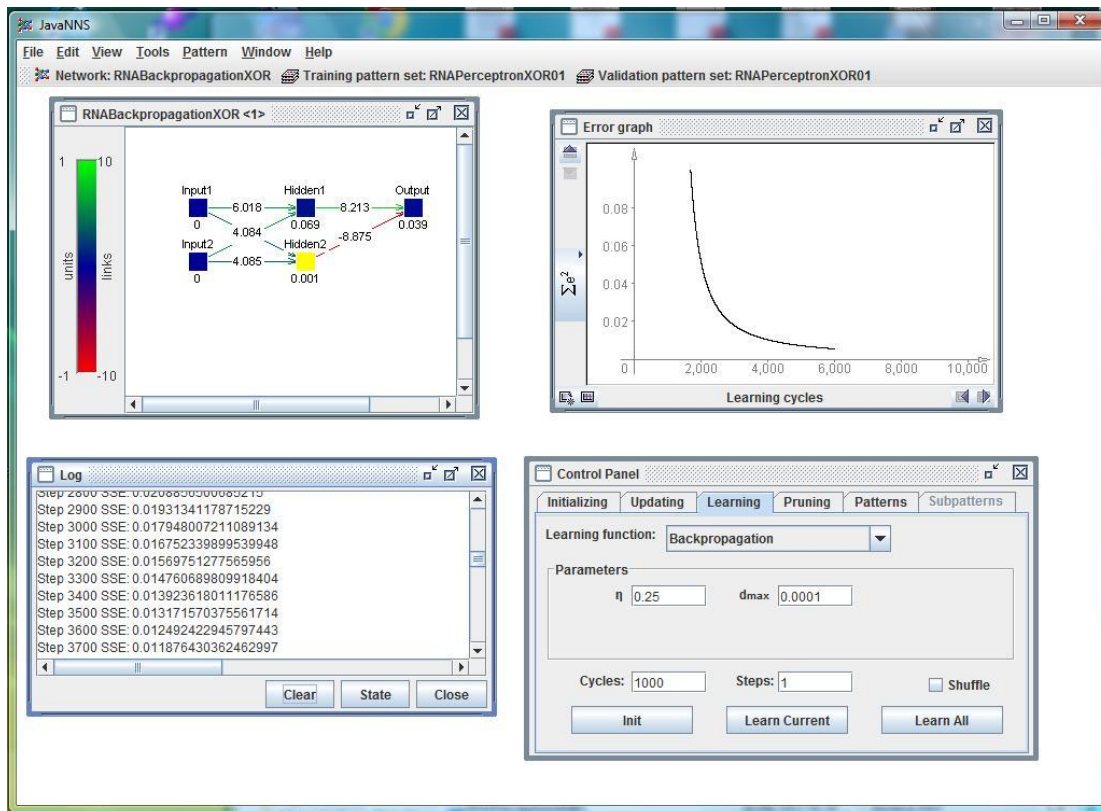
No. of patterns : 4
No. of input units : 2
No. of output units : 1

Input pattern 1:
1 1
Output pattern 1:
0
Input pattern 2:
1 0
Output pattern 2:
1
Input pattern 3:
0 1

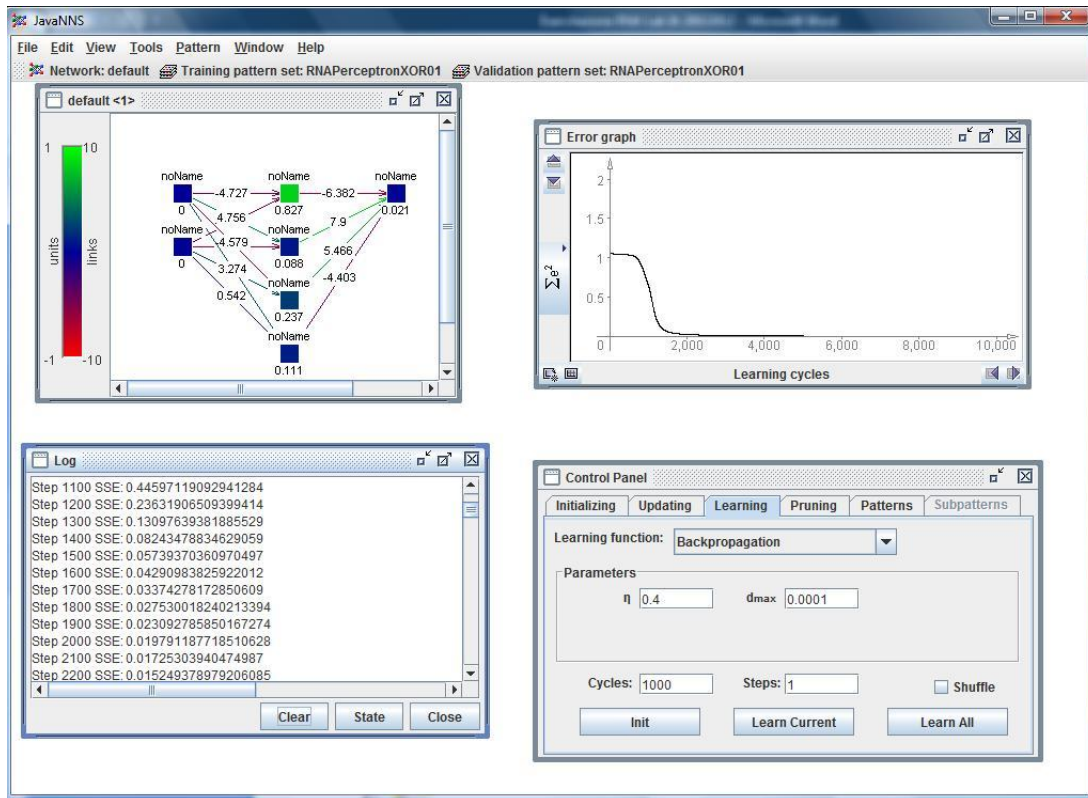
```
# Output pattern 3:
1
# Input pattern 4:
0 0
# Output pattern 4:
0
```

2.3.3. L'implementazione via JavaNNS

Reti multi-strato con 2 neuroni nascosti

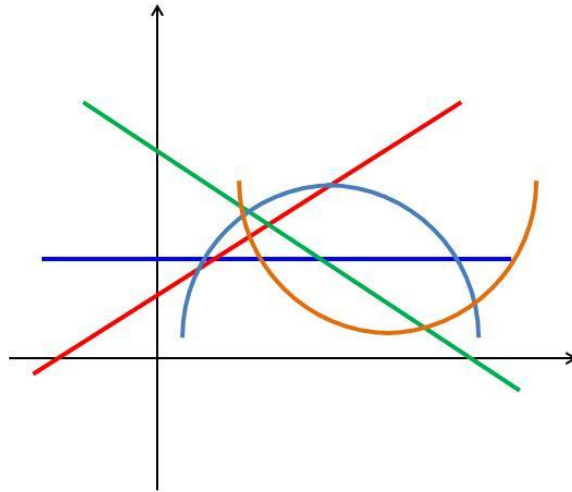


Reti multi-strato con 4 neuroni nascosti



2.4. Classificazione multicategoria: Il problema delle cinque linee

2.4.1. Rappresentazione geometrica del problema



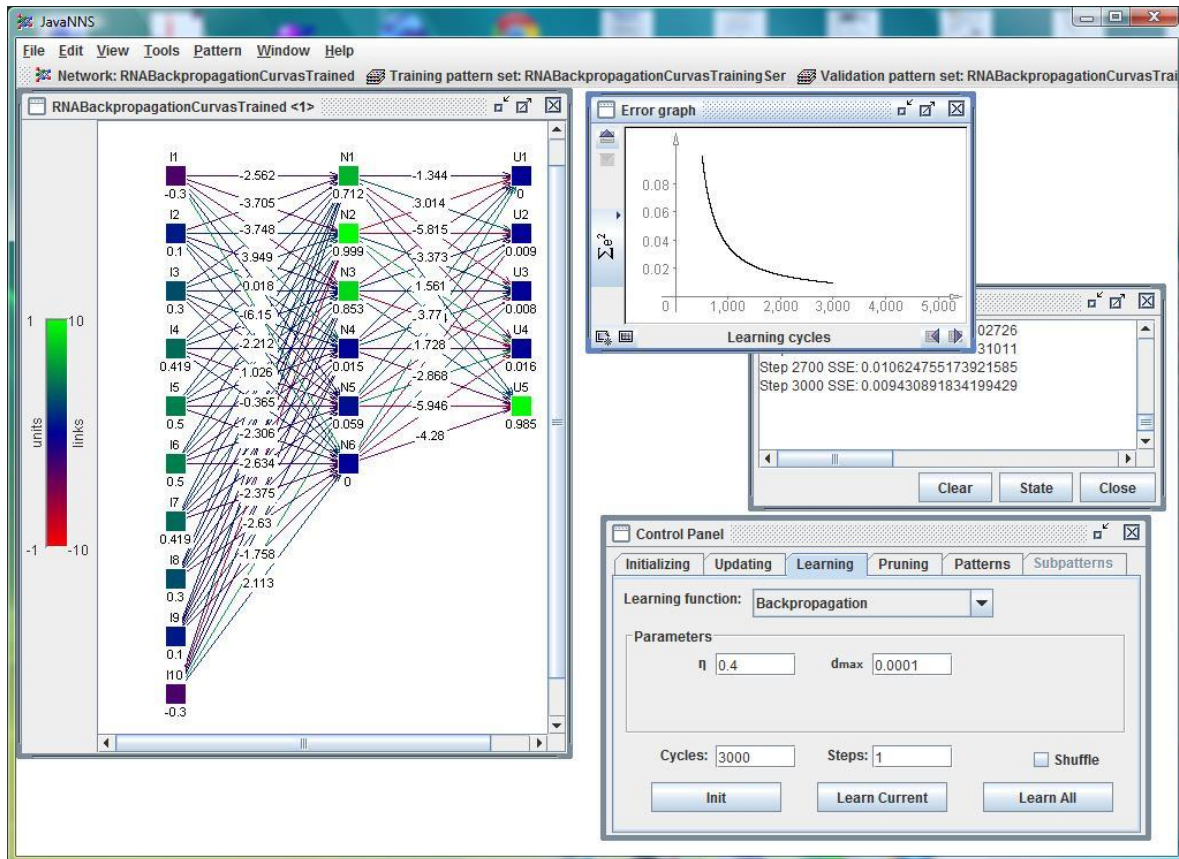
2.4.2. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 20
No. of input units : 10
No. of output units : 5

# Input pattern 1: 0.5 0.28 0.22 0.14 0.1 0.1 0.12 0.18 0.26 0.46 # Output pattern 1: 1 0 0 0 # Input pattern 2: 0.6 0.38 0.32 0.24 0.2 0.2 0.22 0.28 0.36 0.56 # Output pattern 2: 1 0 0 0 # Input pattern 3: 0.7 0.48 0.42 0.34 0.3 0.3 0.32 0.38 0.46 0.66 # Output pattern 3: 1 0 0 0 # Input pattern 4: 0.4 0.18 0.12 0.04 0.0 0.0 0.02 0.08 0.16 0.36 # Output pattern 4: 1 0 0 0 # Input pattern 5: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 # Output pattern 5: 0 1 0 0 # Input pattern 6: 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 # Output pattern 6: 0 1 0 0 # Input pattern 7: 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 # Output pattern 7: 0 1 0 0 # Input pattern 8: -0.1 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 # Output pattern 8: 0 1 0 0 # Input pattern 9: 1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 # Output pattern 9: 0 0 1 0 # Input pattern 10: 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0 -0.1 # Output pattern 10: 0 0 1 0	# Input pattern 11: 0.6 0.5 0.4 0.3 0.2 0.1 0.0 -0.1 -0.2 -0.3 # Output pattern 11: 0 0 1 0 # Input pattern 12: 1.2 1.1 1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 # Output pattern 12: 0 0 1 0 # Input pattern 13: 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 # Output pattern 13: 0 0 0 1 # Input pattern 14: 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 # Output pattern 14: 0 0 0 1 # Input pattern 15: -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 -0.3 # Output pattern 15: 0 0 0 1 # Input pattern 16: -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 # Output pattern 16: 0 0 0 1 # Input pattern 17: 0.1 0.5 0.7 0.8 0.9 0.9 0.82 0.7 0.5 0.1 # Output pattern 17: 0 0 0 1 # Input pattern 18: 0.3 0.7 0.9 1.02 1.1 1.1 1.02 0.9 0.7 0.3 # Output pattern 18: 0 0 0 1 # Input pattern 19: -0.1 0.3 0.5 0.62 0.7 0.7 0.62 0.5 0.3 -0.1 # Output pattern 19: 0 0 0 1 # Input pattern 20: -0.3 0.1 0.3 0.42 0.5 0.5 0.42 0.3 0.1 -0.3 # Output pattern 20: 0 0 0 1
---	---

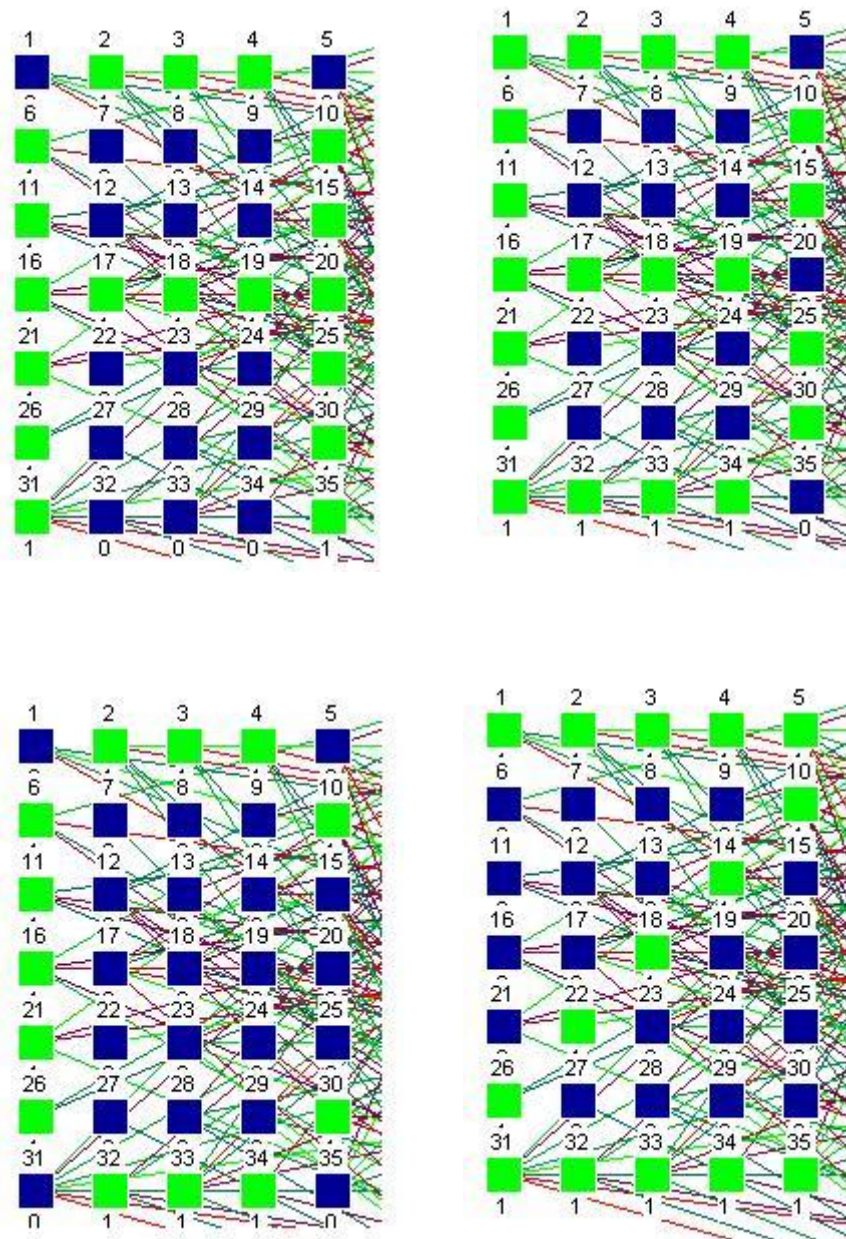
2.4.3. L'implementazione via JavaNNS



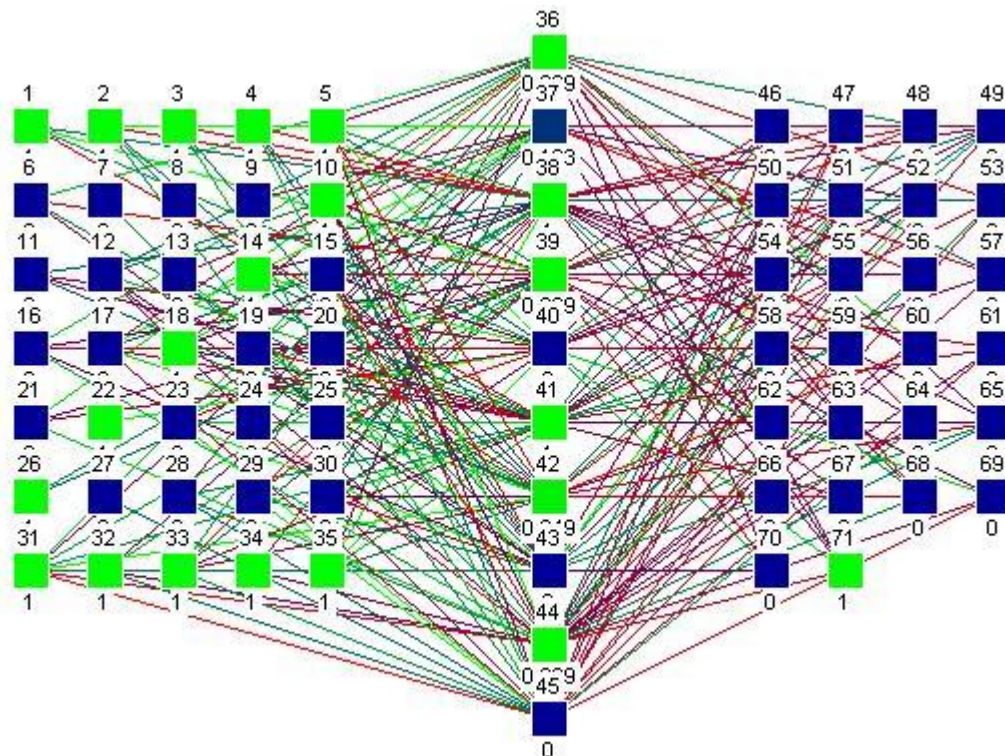
2.5. Riconoscimento di pattern: esempio JavaNNS *letters*

Questo esempio si riferisce a una rete di riconoscimento delle lettere dell'alfabeto (livello molto semplice).

2.5.1. Rappresentazione del problema



2.5.2. Il modello di rete neurale



2.5.3. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Mon Apr 25 18:08:50 1994

No. of patterns : 26
No. of input units : 35
No. of output units : 26

Input pattern 1:

0 1 1 1 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 1
1 0 0 0 1 1 0 0 0 1
1 0 0 0 1

Output pattern 1:

1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0

Input pattern 2:

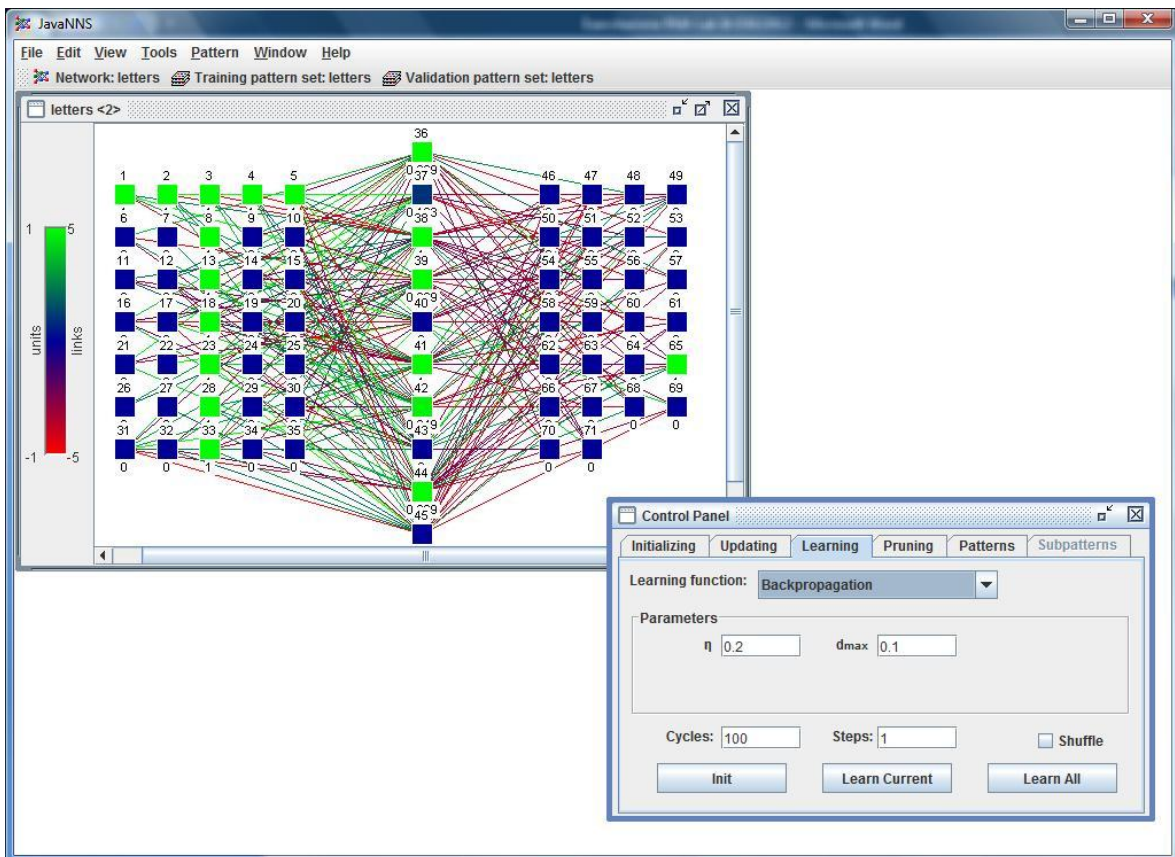
1 1 1 1 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 0
1 0 0 0 1 1 0 0 0 1

```

1 1 1 1 0
# Output pattern 2:
0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
...

```

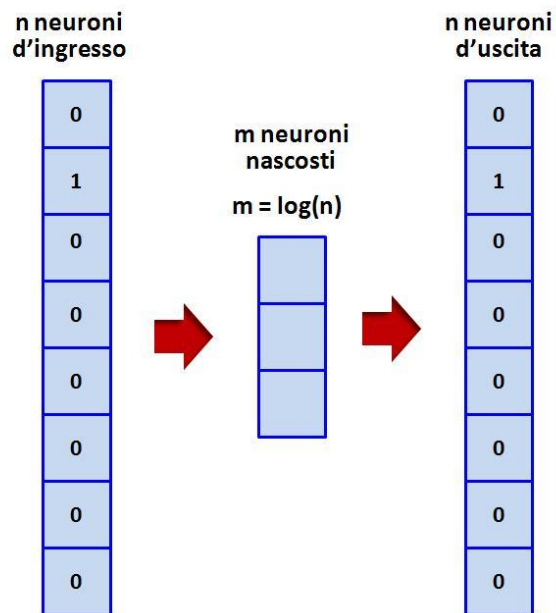
2.5.4. L'implementazione in JavaNNS



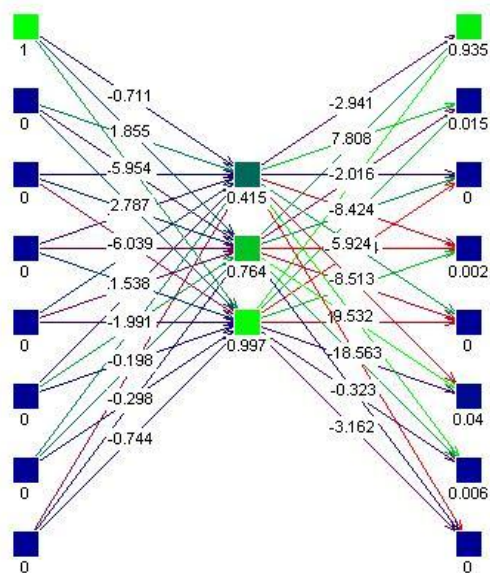
2.6. Codifica-Decodifica di pattern: esempio JavaNNS *encoder*

The task of an n - m - n -encoder network is to code a 1-of- n coding of n input units to $m = \log(n)$ hidden units and to encode the outputs of these m units again to a 1-of- n representation of n units.

2.6.1. Rappresentazione del problema



2.6.2. Il modello di rete neurale



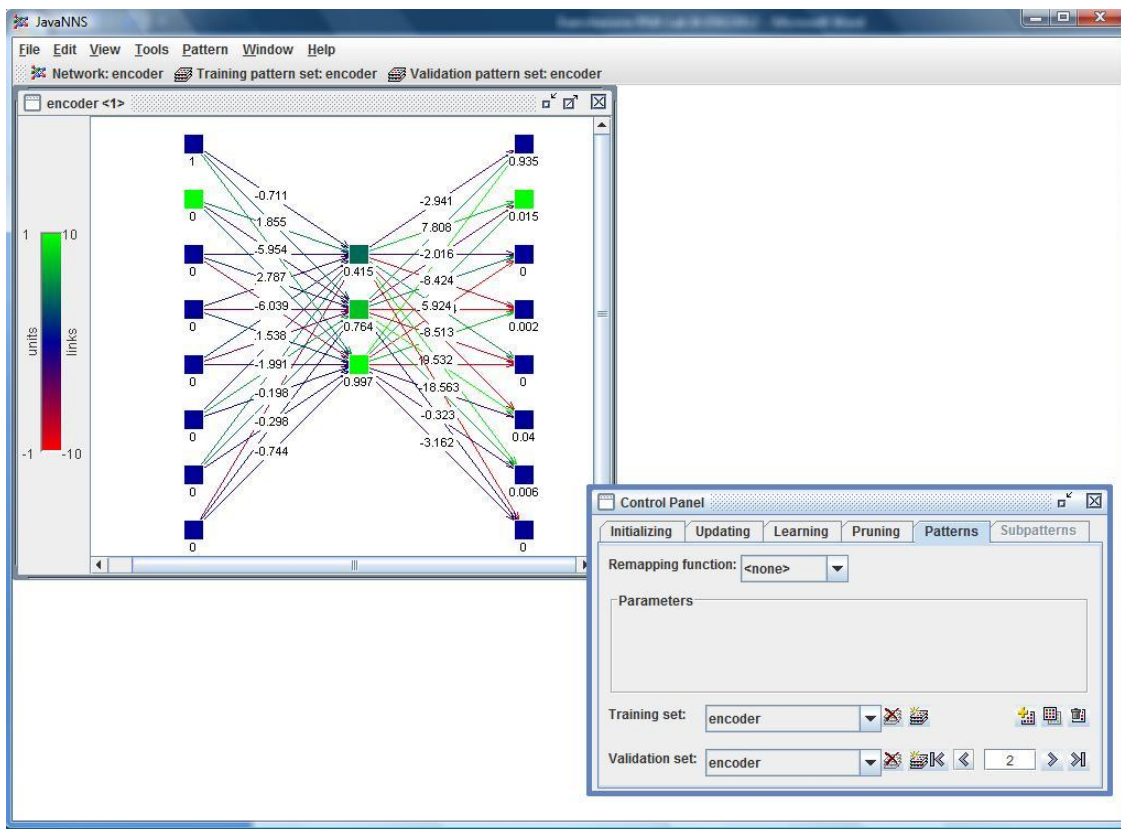
2.6.3. I pattern d'ingresso

SNNS pattern definition file V3.2
generated at Wed May 18 12:01:30 1994

No. of patterns : 8
No. of input units : 8
No. of output units : 8

```
# Input pattern 1:
1 0 0 0 0 0 0 0
# Output pattern 1:
1 0 0 0 0 0 0 0
# Input pattern 2:
0 1 0 0 0 0 0 0
# Output pattern 2:
0 1 0 0 0 0 0 0
# Input pattern 3:
0 0 1 0 0 0 0 0
# Output pattern 3:
0 0 1 0 0 0 0 0
# Input pattern 4:
0 0 0 1 0 0 0 0
# Output pattern 4:
0 0 0 1 0 0 0 0
...
```

2.6.4. L'implementazione in JavaNNS



2.7. Una rete neuronale per la diagnosi di malattie infettive (problema a trattare nell'argomento Sistemi Esperti)

2.7.1. La conoscenza del dominio del problema rappresentata con regole

Le malattie da diagnosticare

- meningitis
- meningism
- pneumonia
- urinary infection
- respiratory failure
- bacterial diarrhea

L'insieme di segni e sintomi

- hiperthermia
- prolonged hyperthermia
- impaired urination
- diarrhea
- cough
- large bowel movements
- frequent bowel movements
- respiratory failure
- alveolar rales
- dyspnea
- tachypnea
- cyanosis
- stiff neck
- meningitis
- kernig meningitis sign
- brudzinski meningitis sign

Le regole per la diagnosi

IF a patient has prolonged hyperthermia AND
a patient has cough AND
a patient has respiratory failure AND
a patient has alveolar rales
THEN the patient's disease is pneumonia

IF a patient has dyspnea AND
a patient has tachypnea AND
a patient has cyanosis
THEN the patient's disease is respiratory failure

If a patient has hyperthermia AND
a patient has cough
THEN the patient's disease is respiratory failure

IF a patient has prolonged hyperthermia AND
a patient has diarrhea AND
a patient has large bowel movements AND
a patient has frequent bowel movements
THEN the patient's disease is bacterial diarrhea

IF a patient has prolonged hyperthermia AND
a patient has impaired urination
THEN the patient's disease is urinary infection

IF a patient has hyperthermia AND
a patient has headache AND
a patient has meningitis sign
THEN the patient's disease is meningitis

IF a patient has stiff neck AND
a patient has the kernig meningitis sign AND
a patient has the brudzinski meningitis sign
THEN the patient's disease is meningism

2.7.2. Rappresentazione del problema per eseguire una diagnosi con un modello di rete neurale

Segni e sintomi	Malattie infettive						
	Pneumonia	Respiratory failure	Respiratory failure	Bacterial diarrhea	Urinary infection	Meningitis	Meningism
iperthermia	1	0	1	1	1	1	0
prolonged hyperthermia	1	0	0	0	1	0	0
impaired urination	0	0	0	0	1	0	0
diarrhea	0	0	0	1	0	0	0
cough	1	0	1	0	0	0	0
large bowel movements	0	0	0	1	0	0	0
frequent bowel movements	0	0	0	1	0	0	0
respiratory failure sign	1	1	1	0	0	0	0
alveolar rales	1	0	0	0	0	0	0
dyspnea	0	1	0	0	0	0	0
tachypnea	0	1	0	0	0	0	0
cyanosis	0	1	0	0	0	0	0
headache	0	0	0	0	0	1	0
stiff neck	0	0	0	0	0	0	1
meningitis sign	0	0	0	0	0	1	0
kernig meningitis sign	0	0	0	0	0	0	1
brudzinski meningitis sign	0	0	0	0	0	0	1

2.7.3. I pattern d'ingresso

**SNNS pattern definition file V3.2
generated at Fri Mar 23 11:00:00 2012**

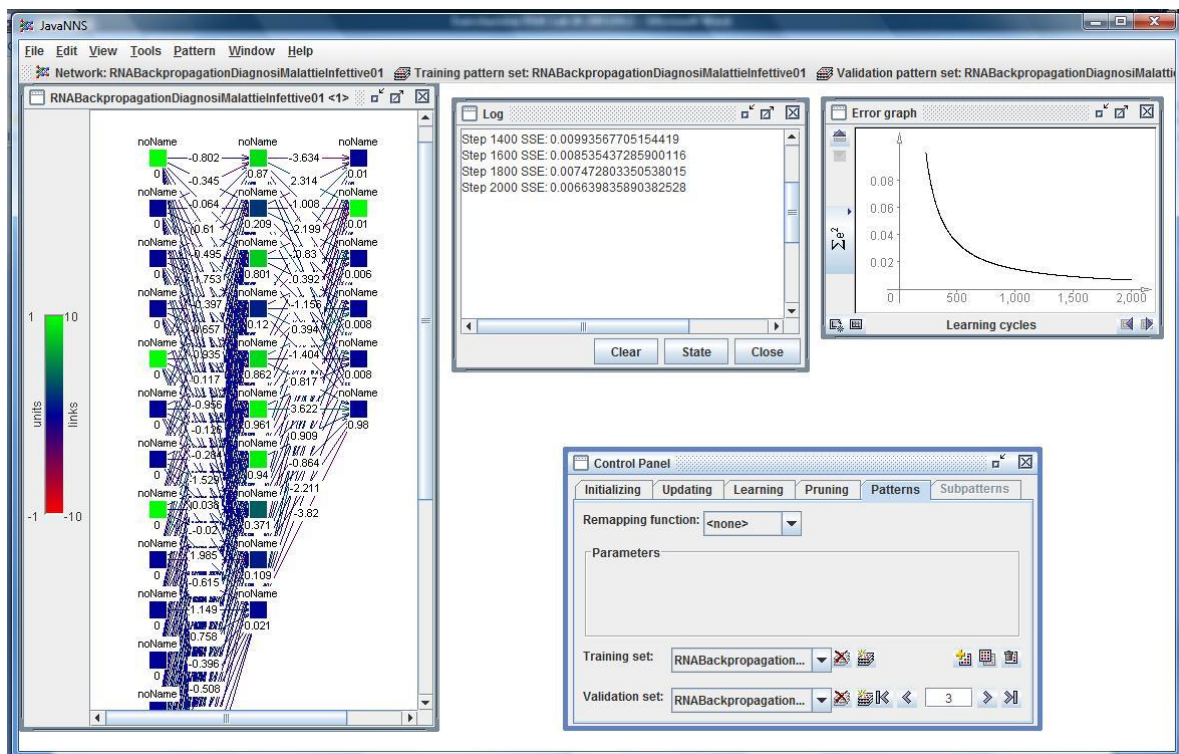
**No. of patterns : 7
No. of input units : 17
No. of output units : 6**

**# Input pattern 1:
1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0
Output pattern 1:
1 0 0 0 0 0
Input pattern 2:
0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0
Output pattern 2:
0 1 0 0 0 0
Input pattern 3:
1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
Output pattern 3:
0 0 1 0 0 0
Input pattern 4:
1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0
Output pattern 4:
0 0 0 1 0 0
Input pattern 5:
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
Output pattern 5:
0 0 0 0 1 0
Input pattern 6:
1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
Output pattern 6:
0 0 0 0 0 1
Input pattern 7:
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
Output pattern 7:
0 0 0 0 0 1**

2.7.4. I pattern di uscita

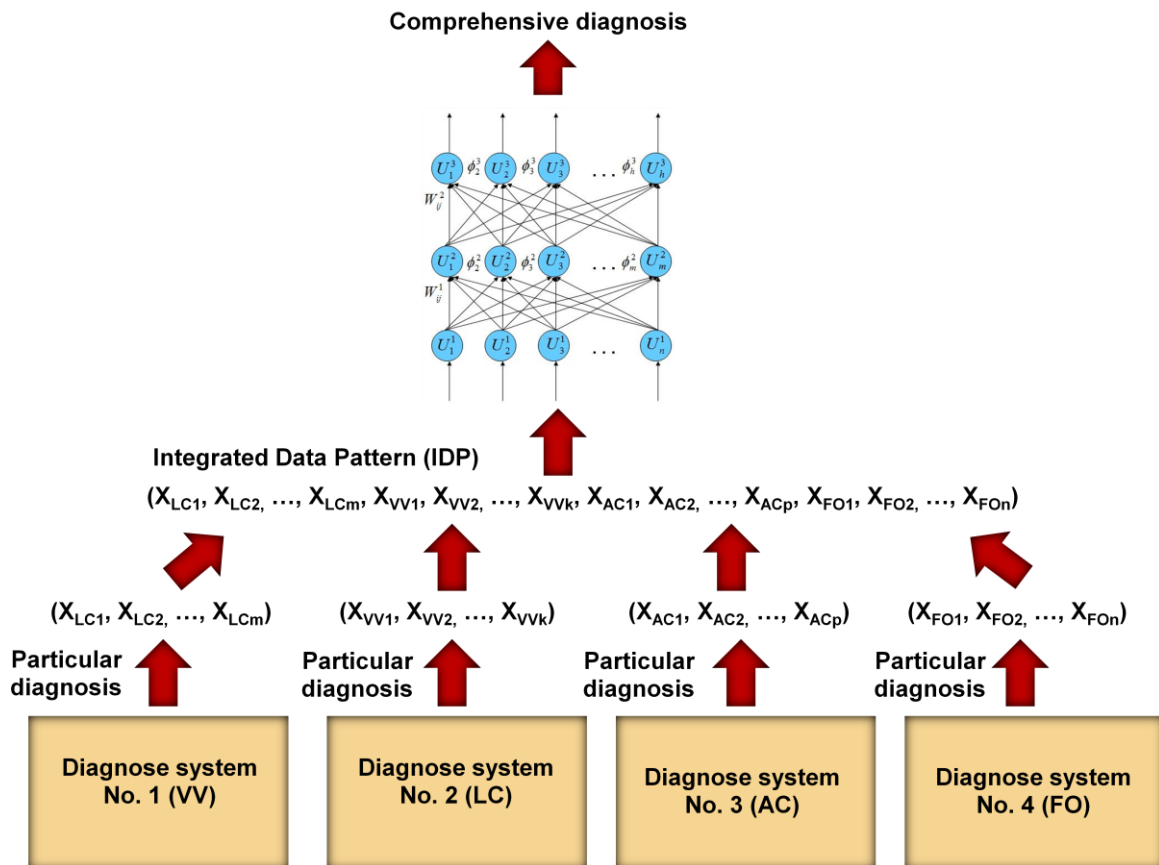
Neurone	Malattia infettiva che rappresenta
1	pneumonia
2	respiratory failure
3	bacterial diarrhea
4	urinary infection
5	meningitis
6	meningism

2.7.5. L'implementazione in JavaNNS

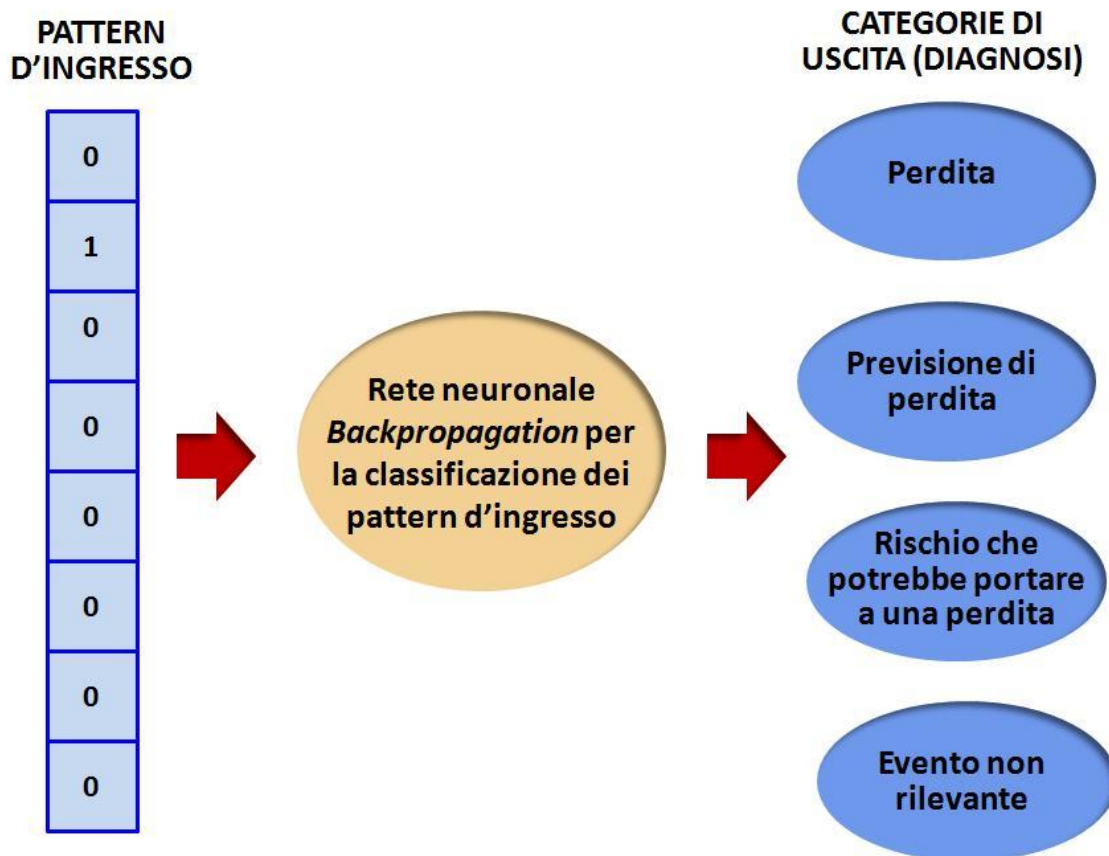


3. Un caso di studio: Rilevamento perdite negli oleodotti (pipeline leak detection)

3.1. Il problema



3.2. Rappresentazione del problema



3.1. I pattern d'ingresso

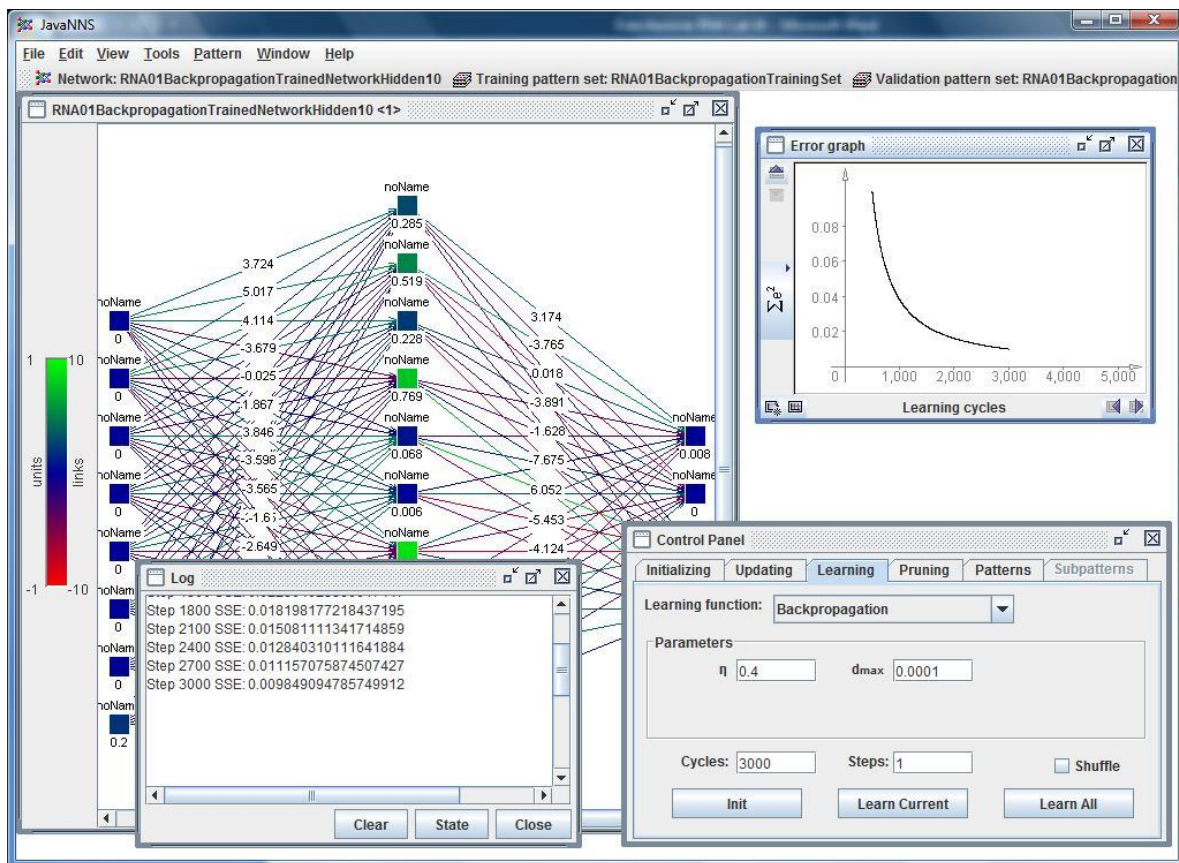
SNNS pattern definition file V3.2
generated at Tue Gen 10 15:50:23 2012

No. of patterns : 56
No. of input units : 8
No. of output units : 4

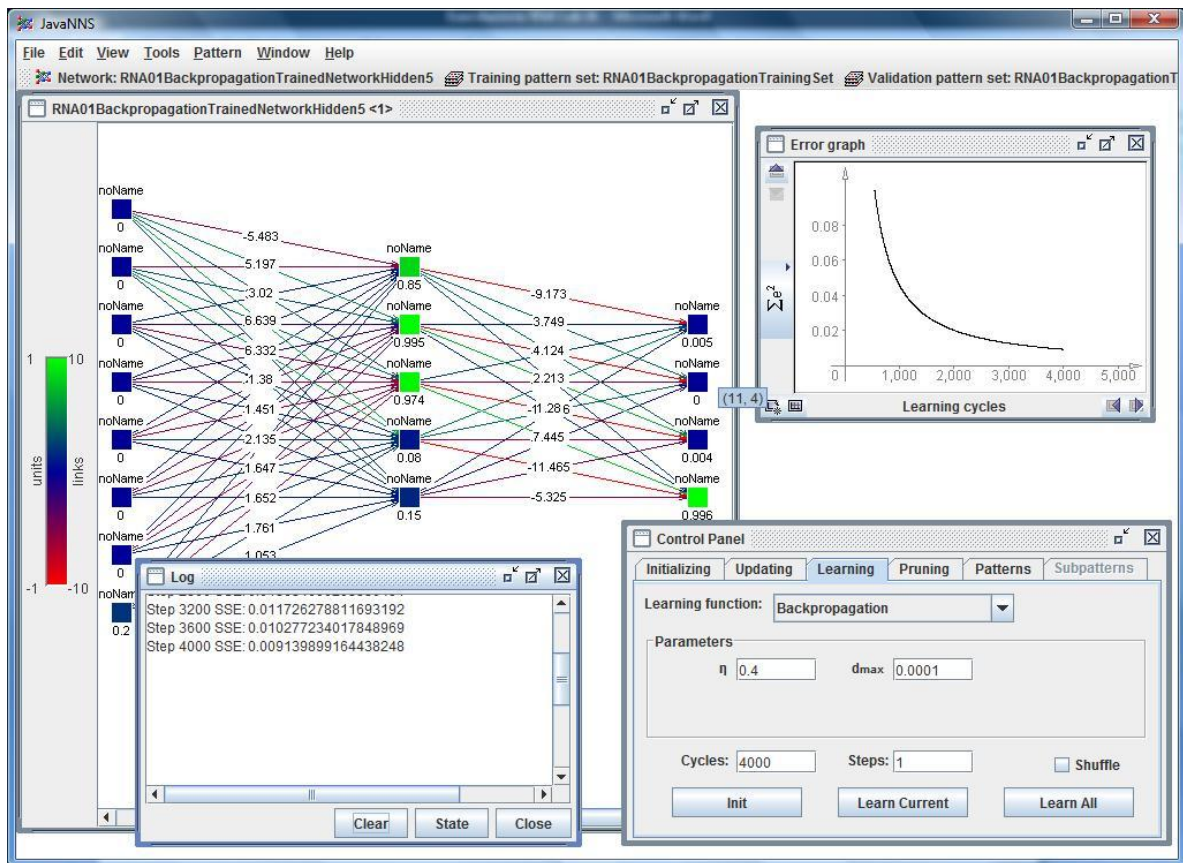
# Input pattern 1: 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 1: 1 0 0 0 # Input pattern 2: 0.9 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 2: 1 0 0 0 # Input pattern 3: 0.8 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 3: 1 0 0 0 # Input pattern 4: 0.6 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 4: 1 0 0 0 # Input pattern 5: 0.9 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 5: 1 0 0 0 # Input pattern 6: 0.6 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 6: 1 0 0 0 # Input pattern 7: 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 7: 1 0 0 0 # Input pattern 8: 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 8: 1 0 0 0 # Input pattern 9: 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 9: 1 0 0 0 # Input pattern 10: 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 10: 1 0 0 0	# Input pattern 11: 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 11: 1 0 0 0 # Input pattern 12: 0.0 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 12: 1 0 0 0 # Input pattern 13: 0.0 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 13: 1 0 0 0 # Input pattern 14: 0.0 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 # Output pattern 14: 1 0 0 0 # Input pattern 15: 1.0 1.0 0.8 0.8 0.8 0.8 0.8 0.8 0.8 # Output pattern 15: 1 0 0 0 # Input pattern 16: 0.9 0.9 0.6 0.6 0.6 0.6 0.6 0.6 0.6 # Output pattern 16: 1 0 0 0 # Input pattern 17: 0.8 0.6 0.8 0.8 0.0 0.8 0.8 0.0 0.0 # Output pattern 17: 1 0 0 0 # Input pattern 18: 0.6 0.8 0.0 0.8 0.8 0.0 0.8 0.9 0.9 # Output pattern 18: 1 0 0 0 # Input pattern 19: 0.9 0.6 0.6 0.0 0.6 0.6 0.0 0.6 0.6 # Output pattern 19: 1 0 0 0 # Input pattern 20: 0.6 0.9 0.0 0.8 0.0 0.0 0.8 0.0 0.0 # Output pattern 20: 1 0 0 0	# Input pattern 21: 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 # Output pattern 21: 0 1 0 0 # Input pattern 22: 0.0 0.0 0.8 0.8 0.8 0.8 0.8 0.8 0.8 # Output pattern 22: 0 1 0 0 # Input pattern 23: 0.0 0.0 0.9 0.9 0.0 0.9 0.9 0.0 0.0 # Output pattern 23: 0 1 0 0 # Input pattern 24: 0.0 0.0 0.0 0.9 0.9 0.0 0.9 0.9 0.9 # Output pattern 24: 0 1 0 0 # Input pattern 25: 0.0 0.0 0.9 0.0 0.9 0.9 0.0 0.9 0.9 # Output pattern 25: 0 1 0 0 # Input pattern 26: 0.0 0.0 0.9 0.9 0.0 0.0 0.9 0.9 0.9 # Output pattern 26: 0 1 0 0 # Input pattern 27: 0.0 0.0 0.0 0.9 0.9 0.9 0.9 0.0 0.0 # Output pattern 27: 0 1 0 0 # Input pattern 28: 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 # Output pattern 28: 0 1 0 0	# Input pattern 29: 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 # Output pattern 29: 0 1 0 0 # Input pattern 30: 0.0 0.0 0.9 0.9 0.9 0.0 0.0 0.0 0.0 # Output pattern 30: 0 1 0 0 # Input pattern 31: 0.0 0.0 0.0 0.0 0.0 0.9 0.9 0.9 0.9 # Output pattern 31: 0 1 0 0 # Input pattern 32: 0.0 0.0 0.7 0.7 0.7 0.7 0.7 0.7 0.7 # Output pattern 32: 0 1 0 0 # Input pattern 33: 0.0 0.0 0.6 0.6 0.0 0.0 0.0 0.0 0.0 # Output pattern 33: 0 0 1 0 # Input pattern 34: 0.0 0.0 0.0 0.6 0.6 0.0 0.0 0.0 0.0 # Output pattern 34: 0 0 1 0 ...
--	--	--	---

3.2. La soluzione via JavaNNS

Rete neurale Backpropagation 8-10-4



Rete neurale Backpropagation 8-5-4



4. Progetto finale per il tema Reti Neurali Artificiali

Formare dei piccoli gruppi di lavoro.

Scegliere uno dei seguenti argomenti:

- Classificazione/riconoscimento di pattern in scienze biomediche (sequenze aminoacidiche, sequenze polipeptidiche, pattern di espressione genica, ecc.).
- Diagnosi di malattie in un dominio medico specifico (cardiovascolare, apparato respiratorio, sistema digerente, diabete, cancro, ecc.).
- Diagnosi e soluzioni dei problemi o malfunzionamenti di un sistema elettronico (circuito integrato, server, rete di computer, rete wireless, rete di sensori, ecc.).
- Riconoscimento di caratteri.
- Codifica e decodifica di segnali.

Per l'argomento scelto realizzare una proposta di progetto che consideri:

- i. Descrizione e rappresentazione del problema.
- ii. Modello di rete neurale artificiale proposto.
- iii. Pattern d'ingresso e pattern di uscita.
- iv. Soluzione via JavaNNS (file .net e .pat).