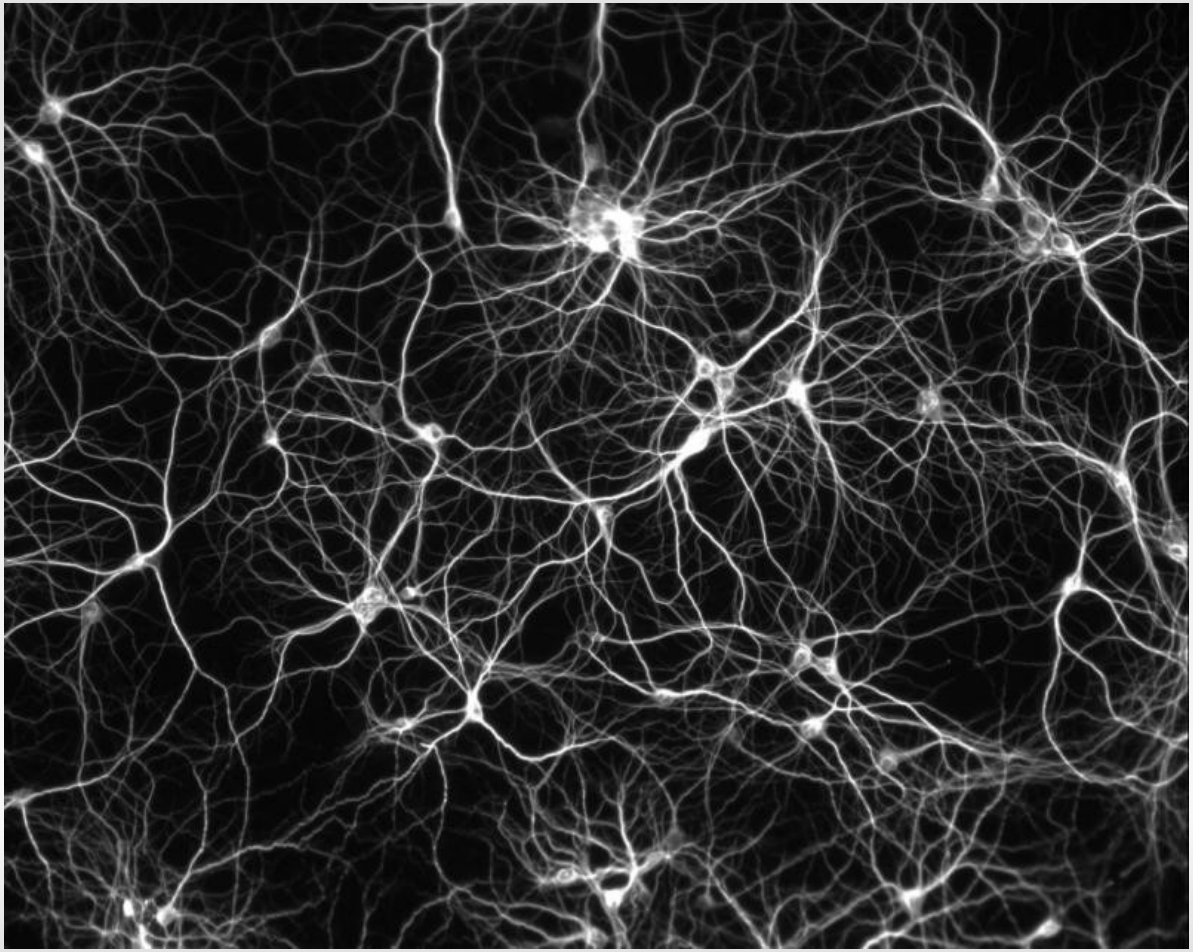
Reti neurali artificiali



Chi sono?

- Daniele Iamartino (aka Otacon22)
- Membro del ”team” HgComo
- Studente/sperimentatore
- Futura matricola del PoliMi, ex liceale con tesina sulle reti neurali!

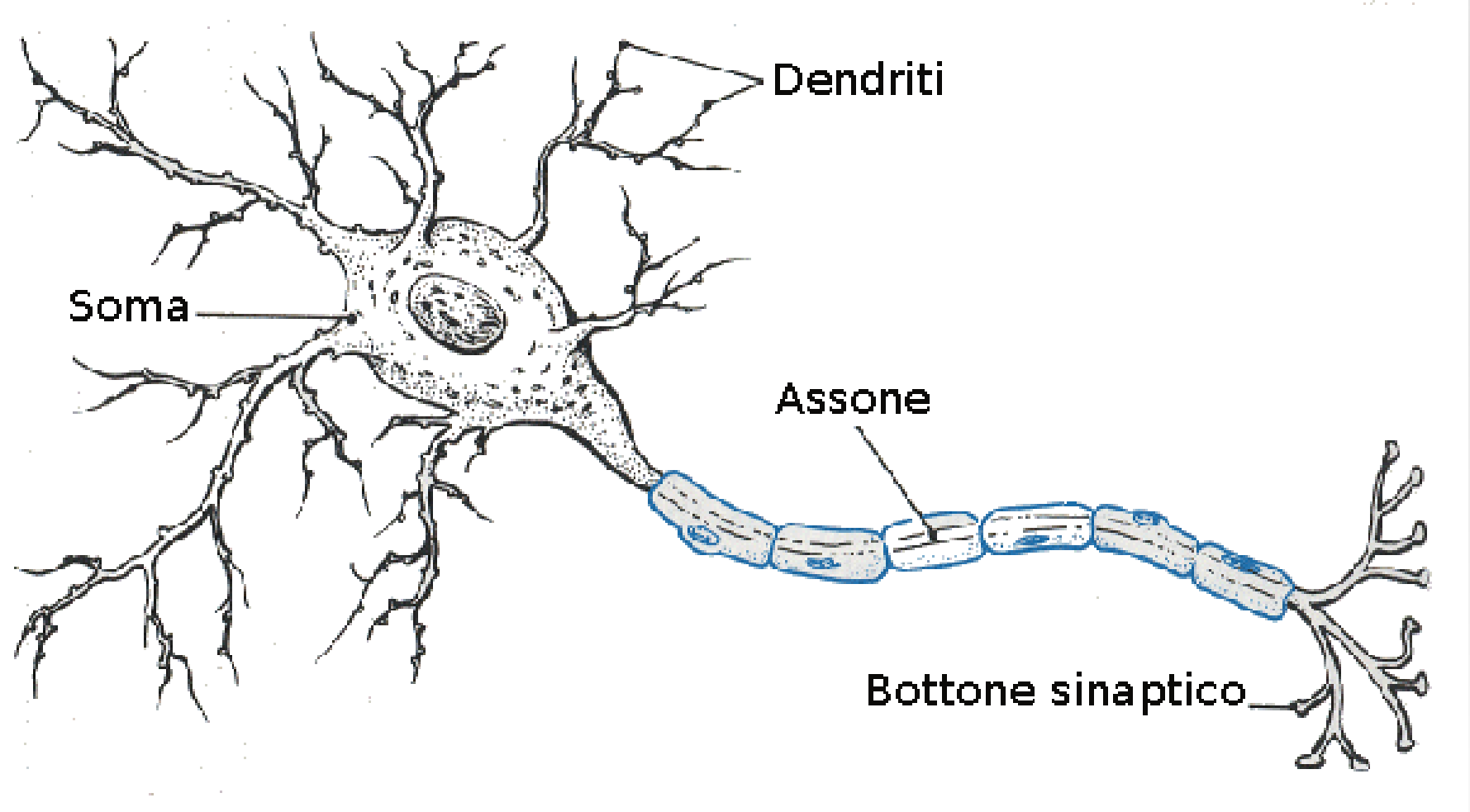
Chi NON sono!

- Un esperto di reti neurali artificiali

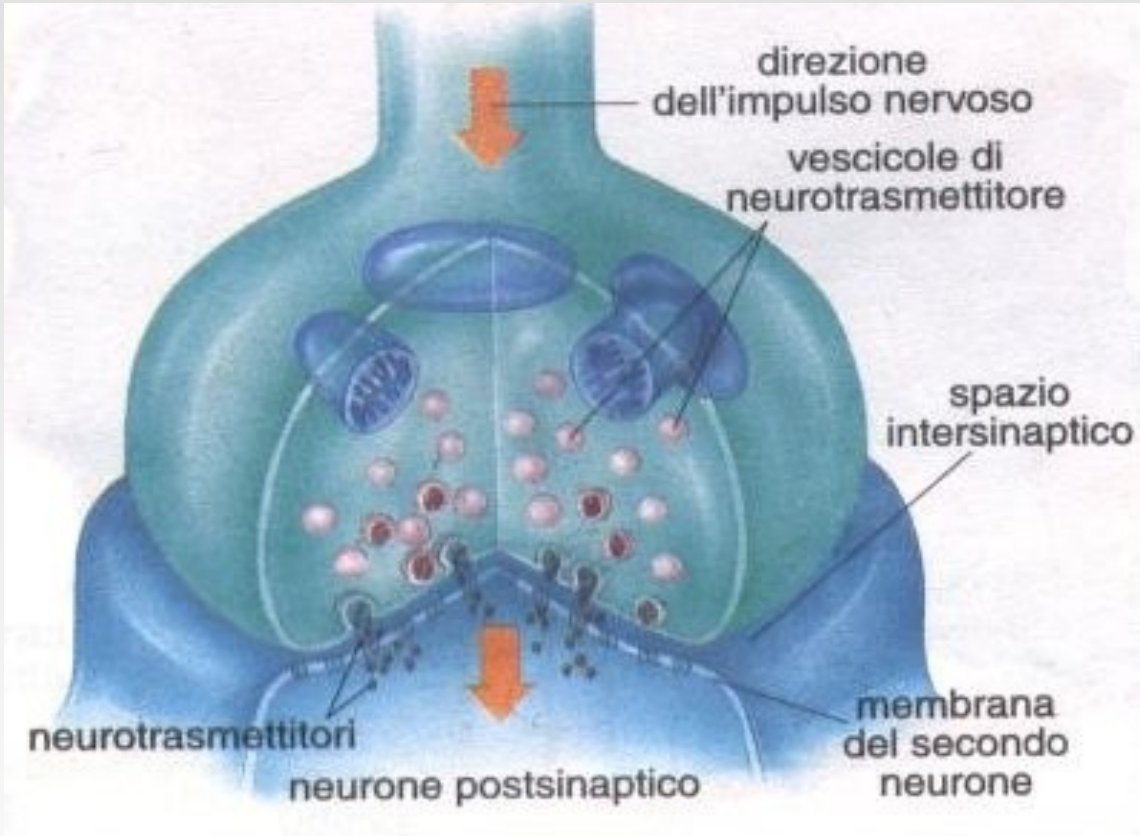
Un po' di storia

- 1936 - Turing e il primo modello di macchina calcolatrice programmabile
- 1943 – McCulloch e Pitts propongono un primo modello di neurone artificiale e la sua conessione in reti
- 1970-1980 - Interruzione dei finanziamenti alle ricerche
- 1985 – Rumelhart pubblica il nuovo algoritmo di ”retro-propagazione dell'errore”

Il neurone biologico

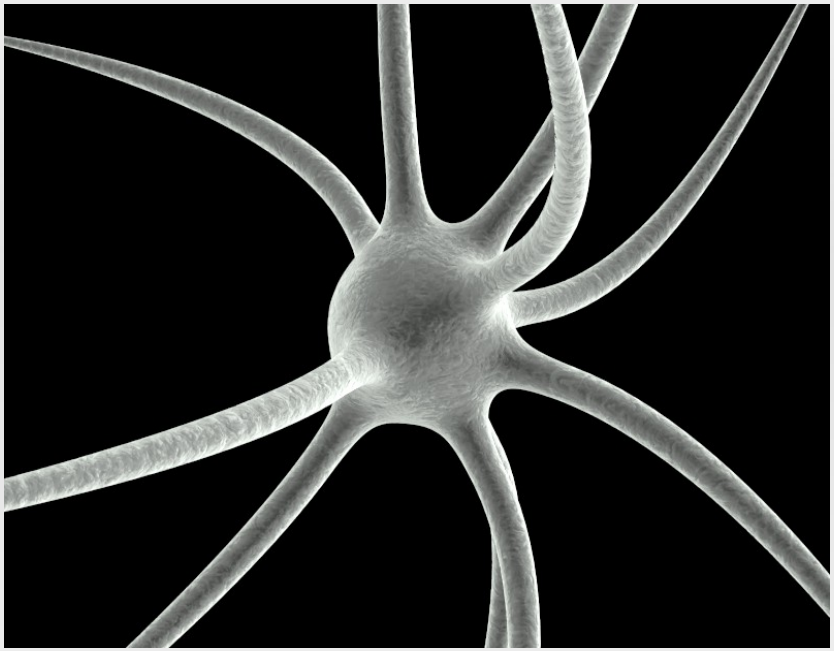


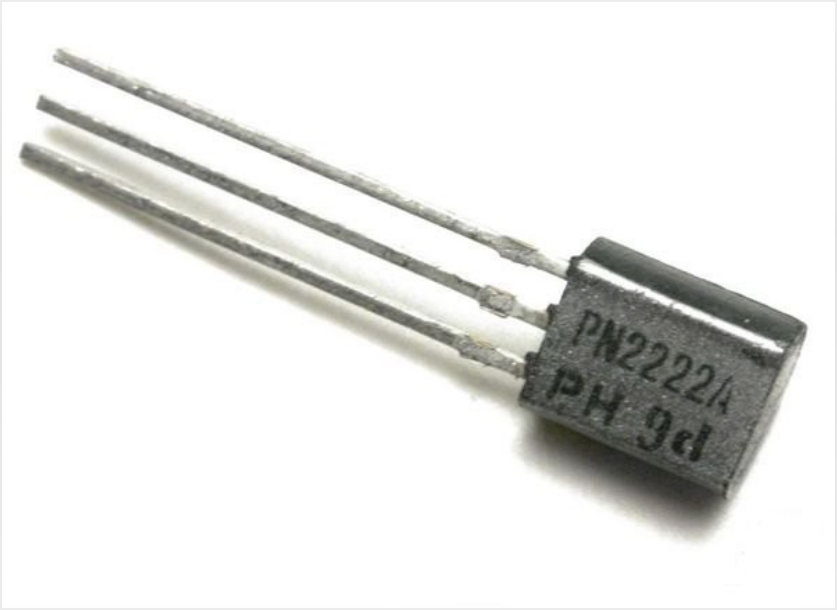
La sinapsi

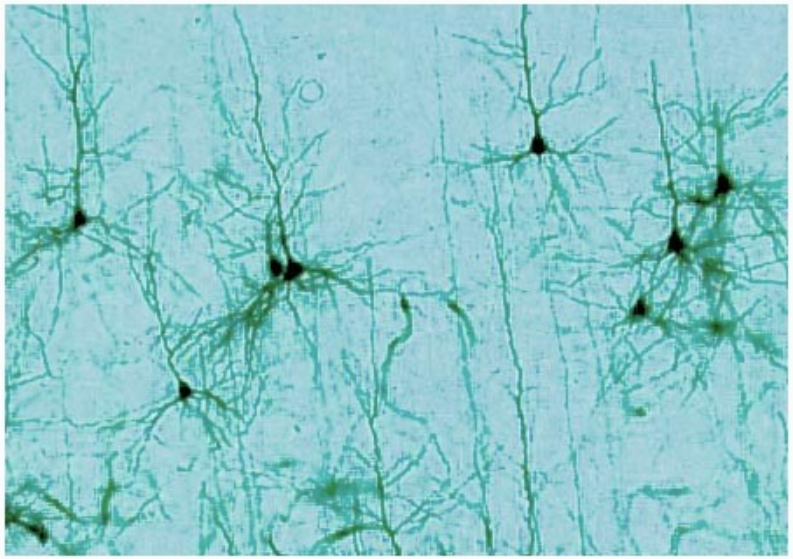


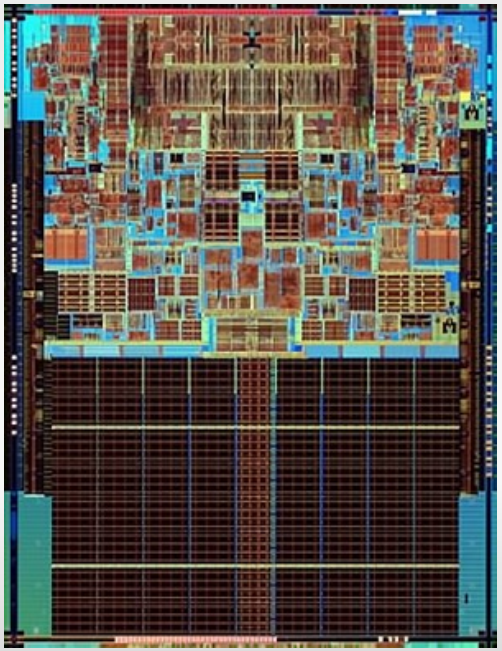
- La sinapsi funziona da ”interfaccia” tra due neuroni
- Può amplificare o smorzare il segnale che la attraversa
- Funziona con un meccanismo elettrochimico

Confronti



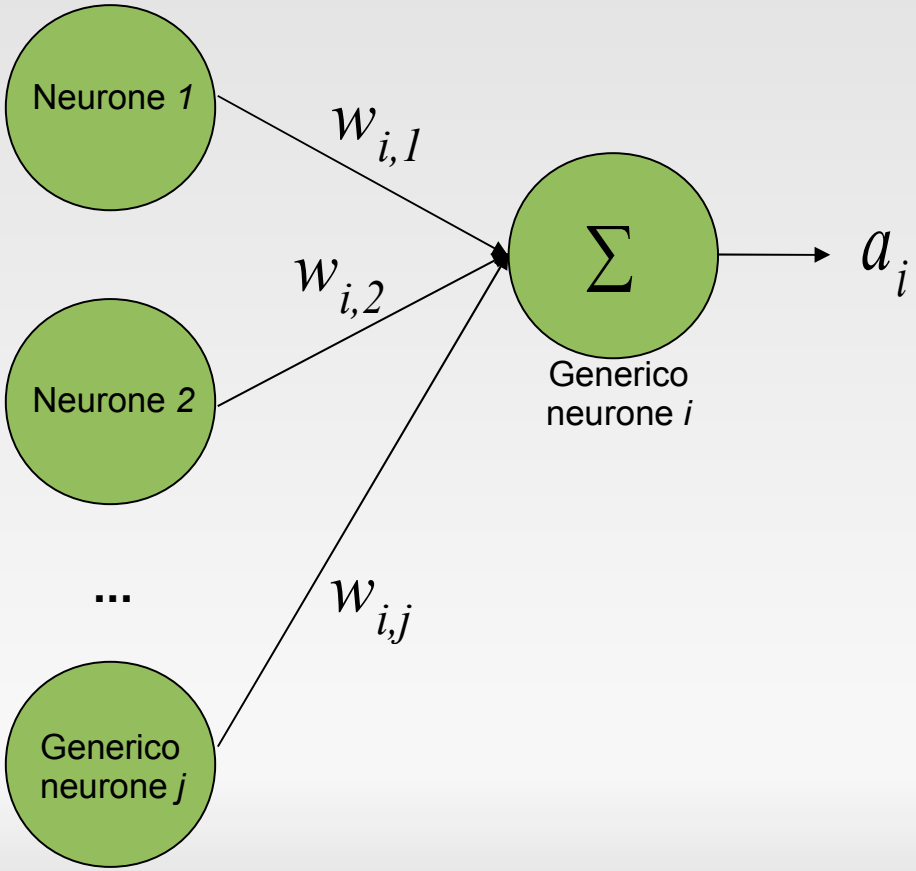






Valore di attivazione

- Si indica il *valore di attivazione* con a_i
- Il peso sinaptico si indica invece con $w_{i,j}$, ovvero il valore del collegamento tra un neurone *i* ed un neurone *j*



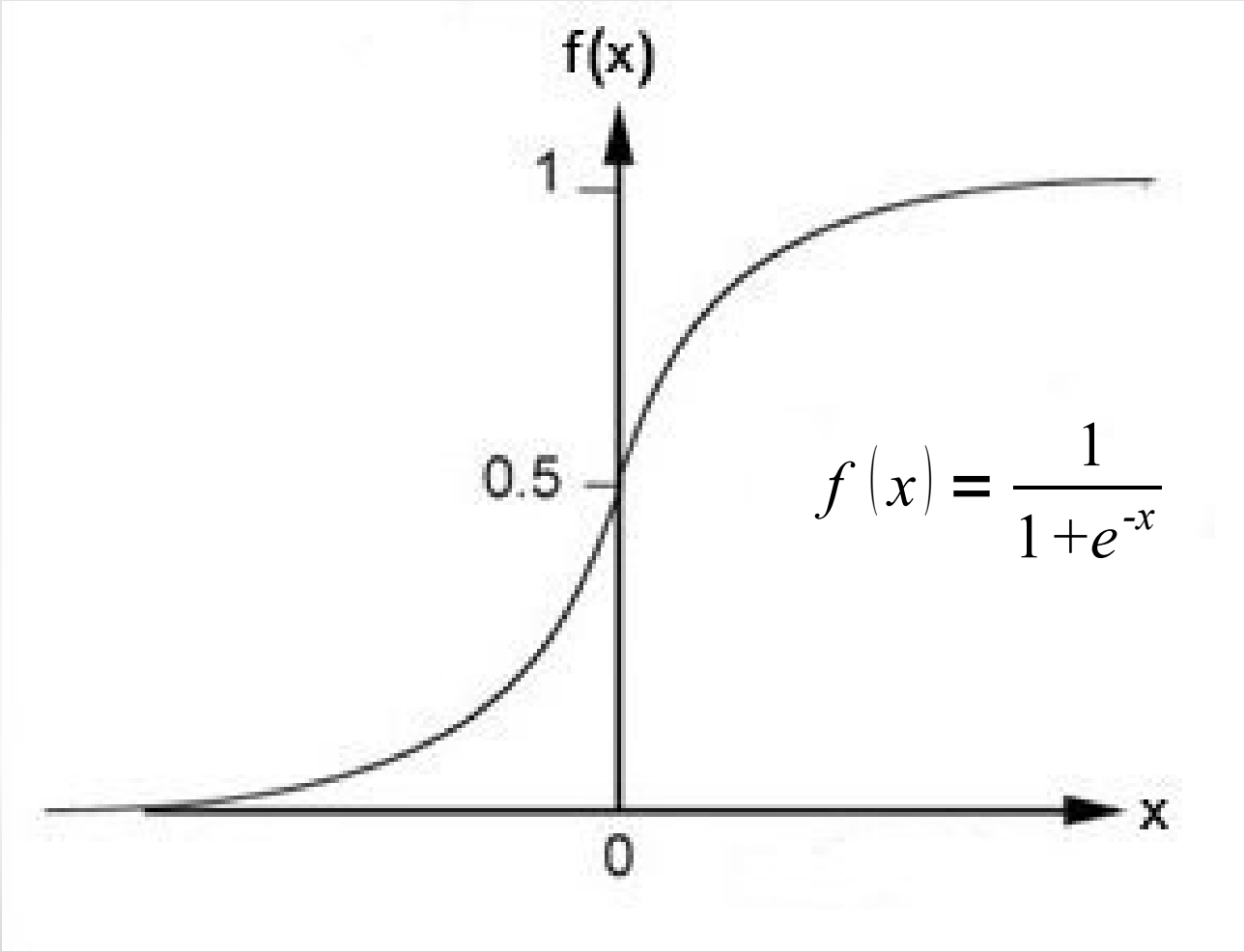
$$a_i = f\left(\sum_j w_{i,j} \cdot a_i\right)$$

Funzione di trasferimento

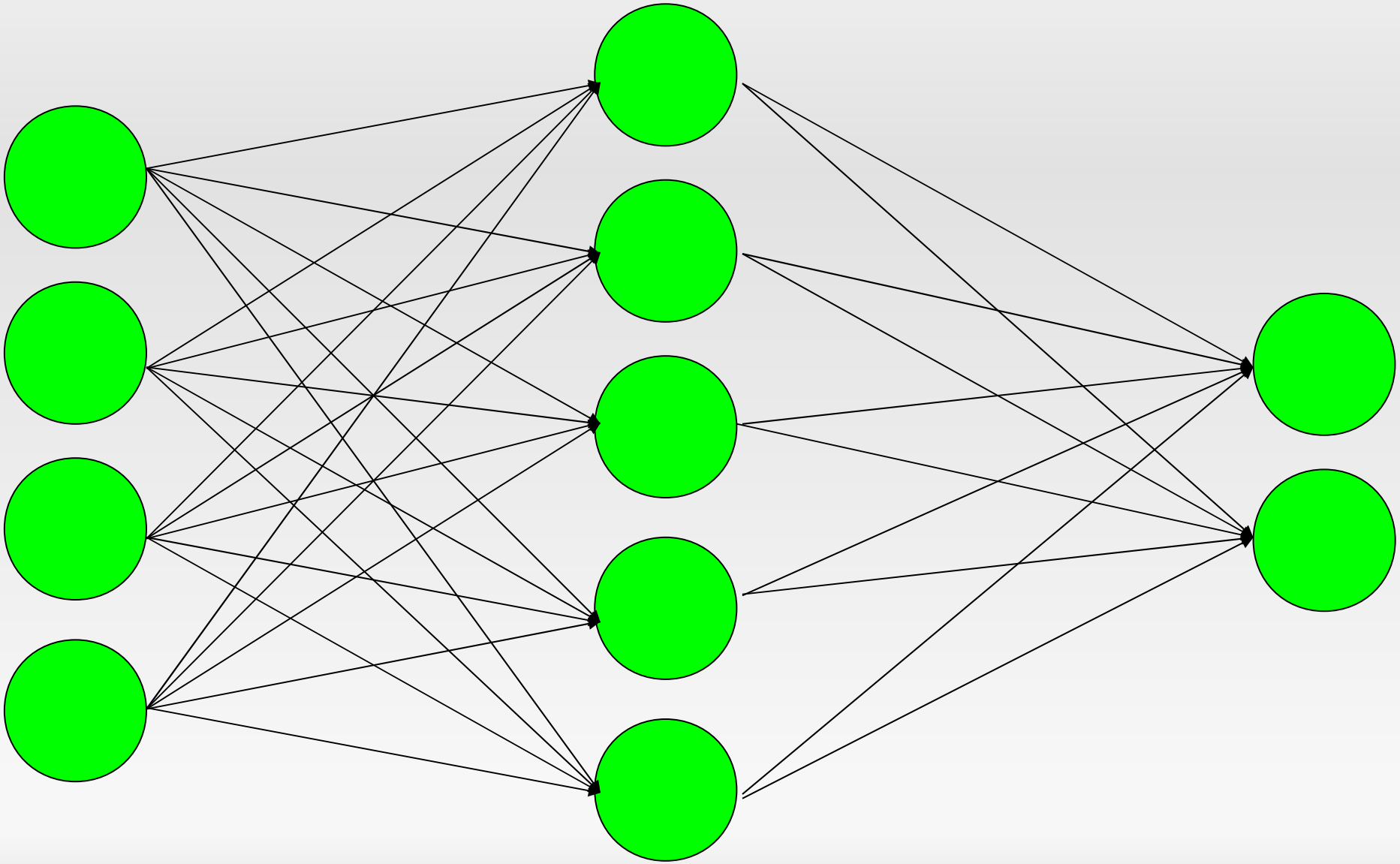
Somma pesata

Funzione di trasferimento

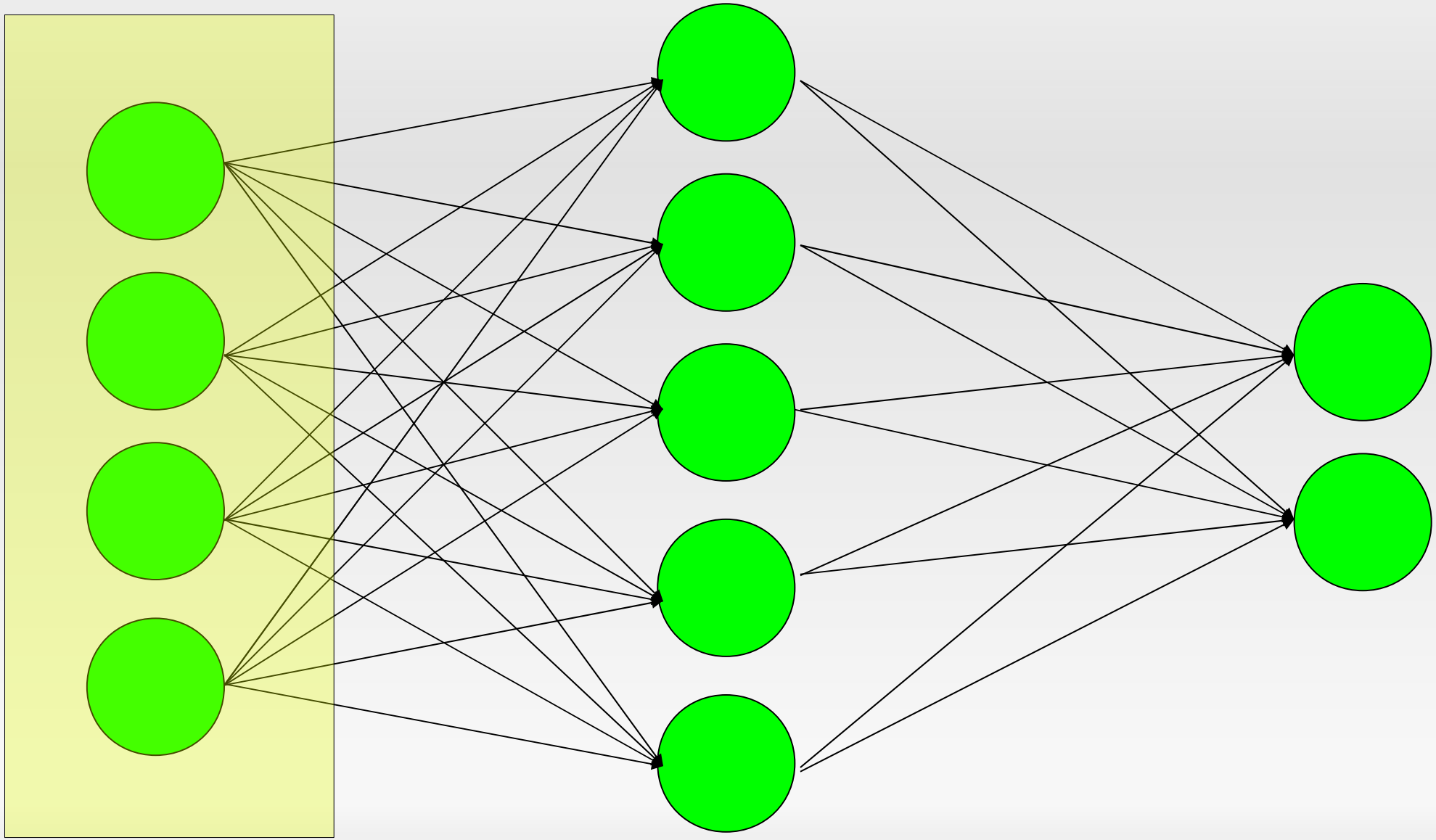
- La più utilizzata è la *sigmoide logistica* o *funzione sigmoidale*
  - *Dominio: tutti i numeri reali*
  - *Codominio: compreso in* $(0\,;1)$



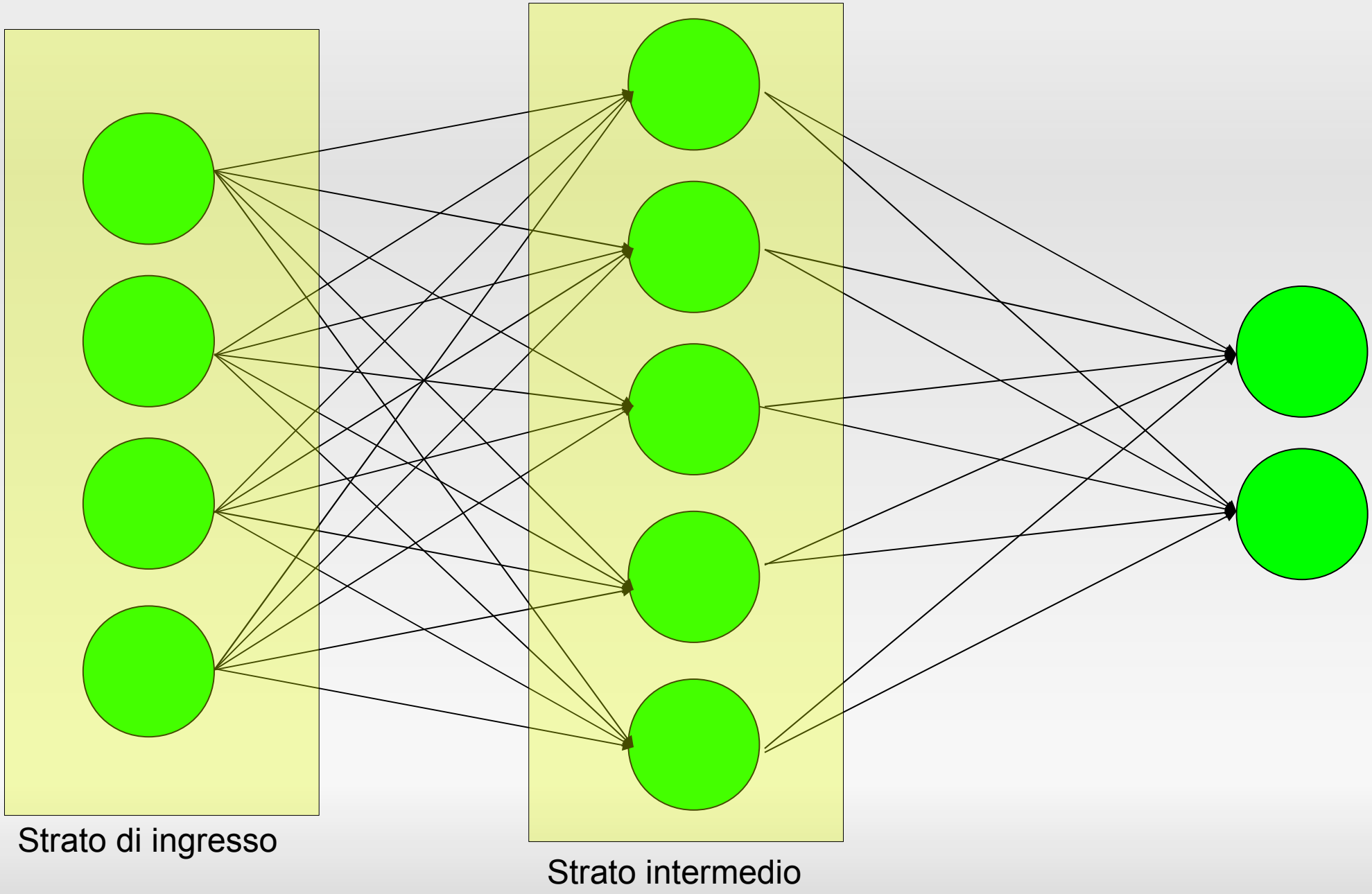
La rete *feed-forward*



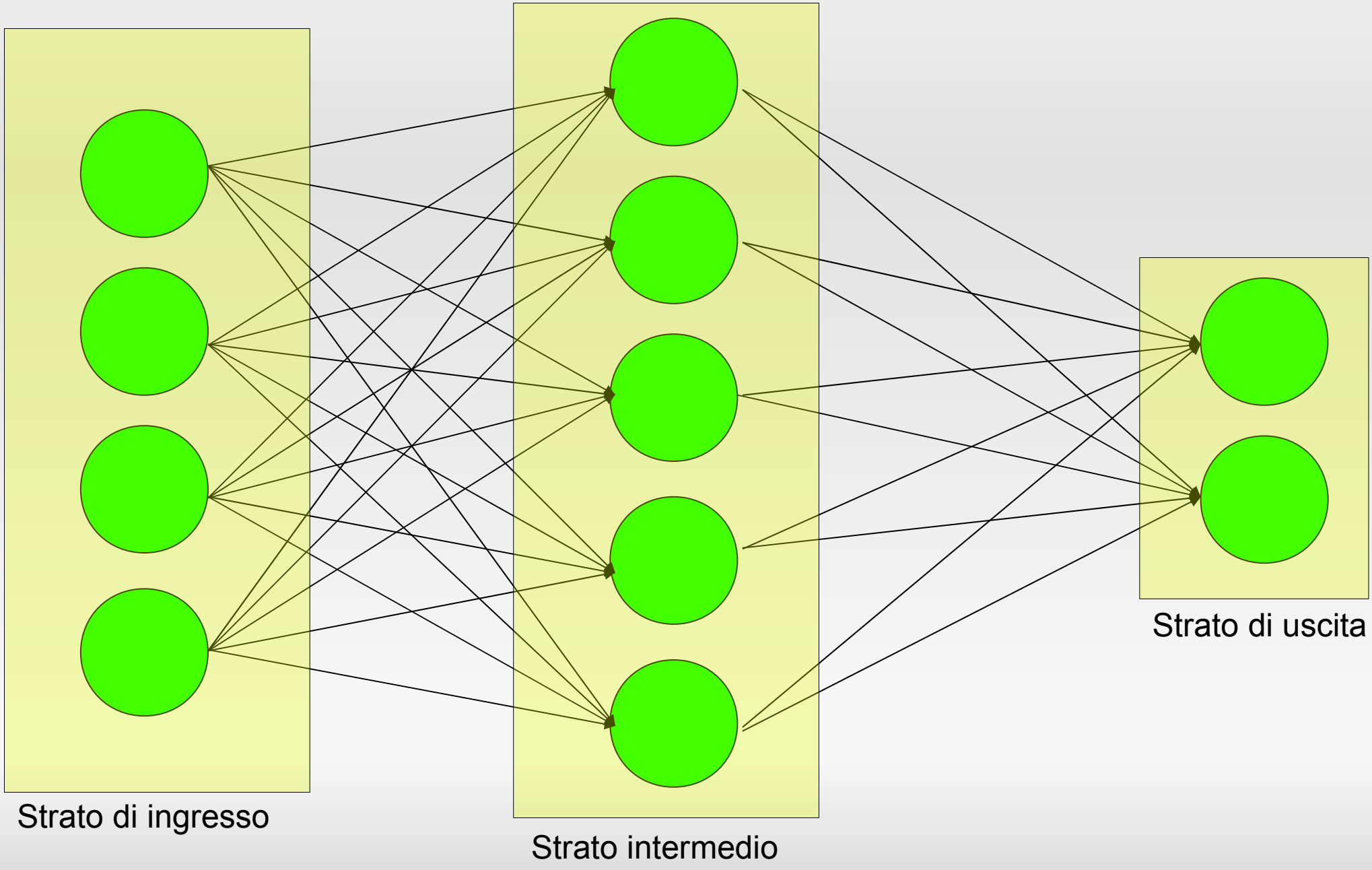
La rete *feed-forward*



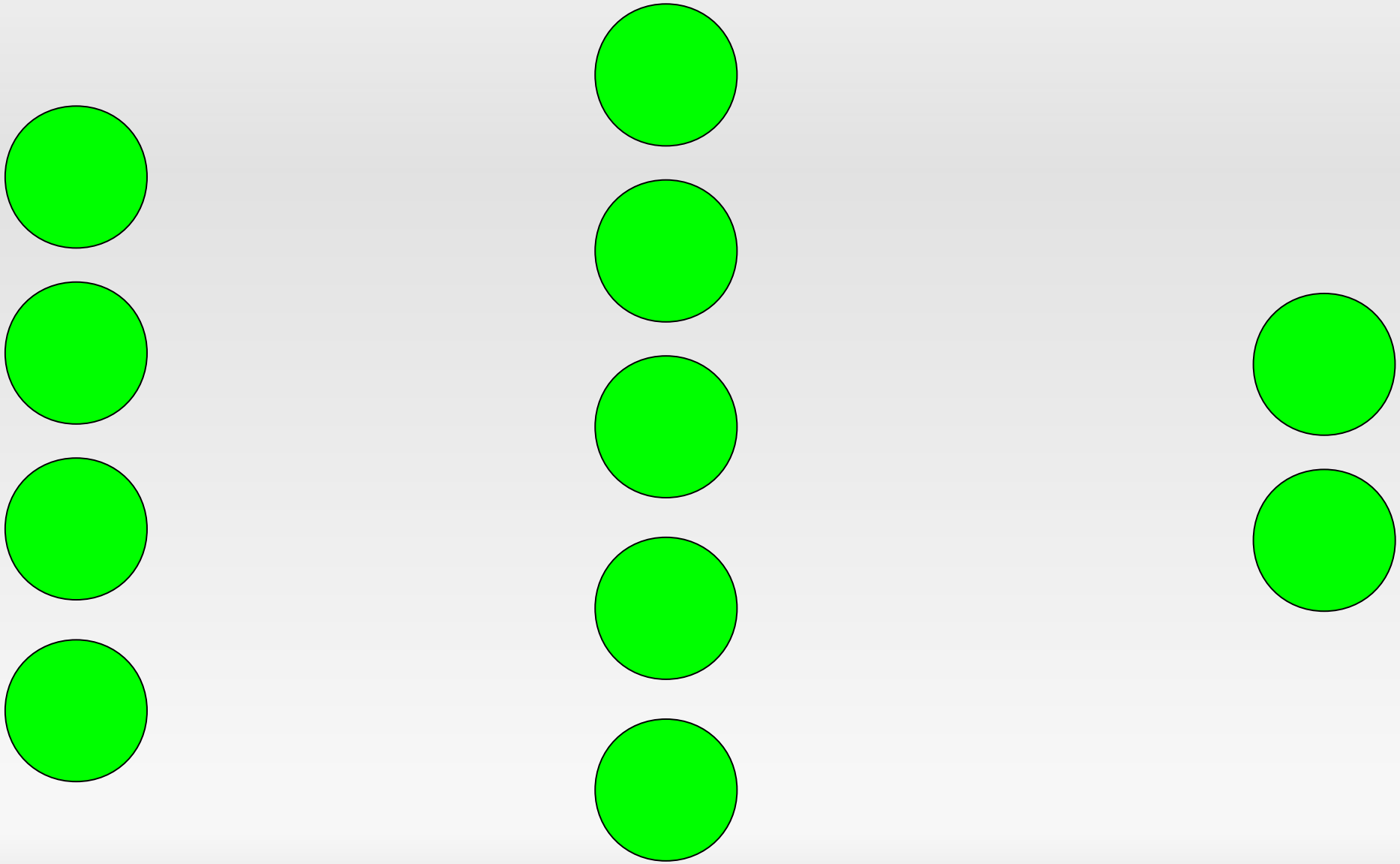
La rete *feed-forward*



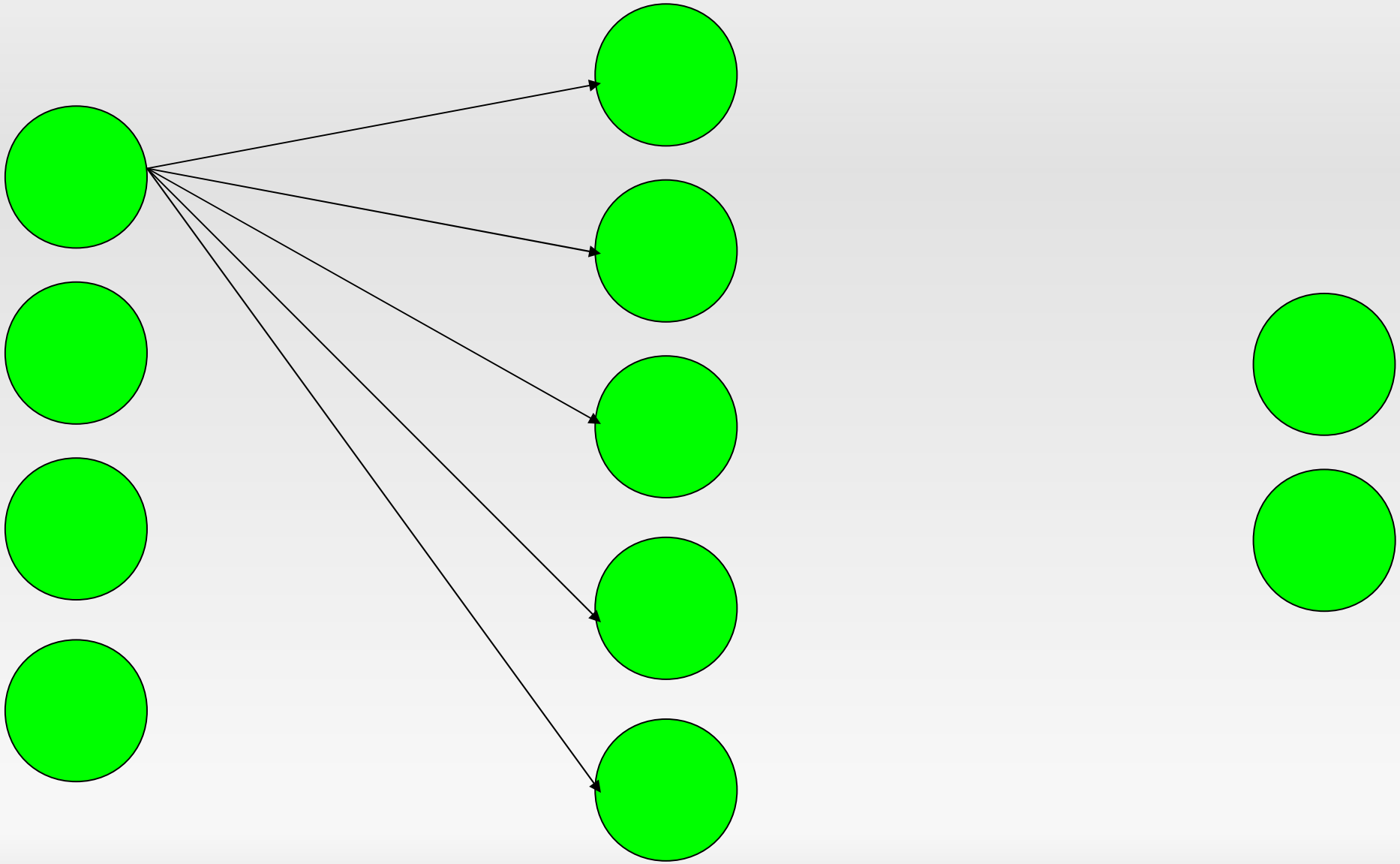
La rete *feed-forward*



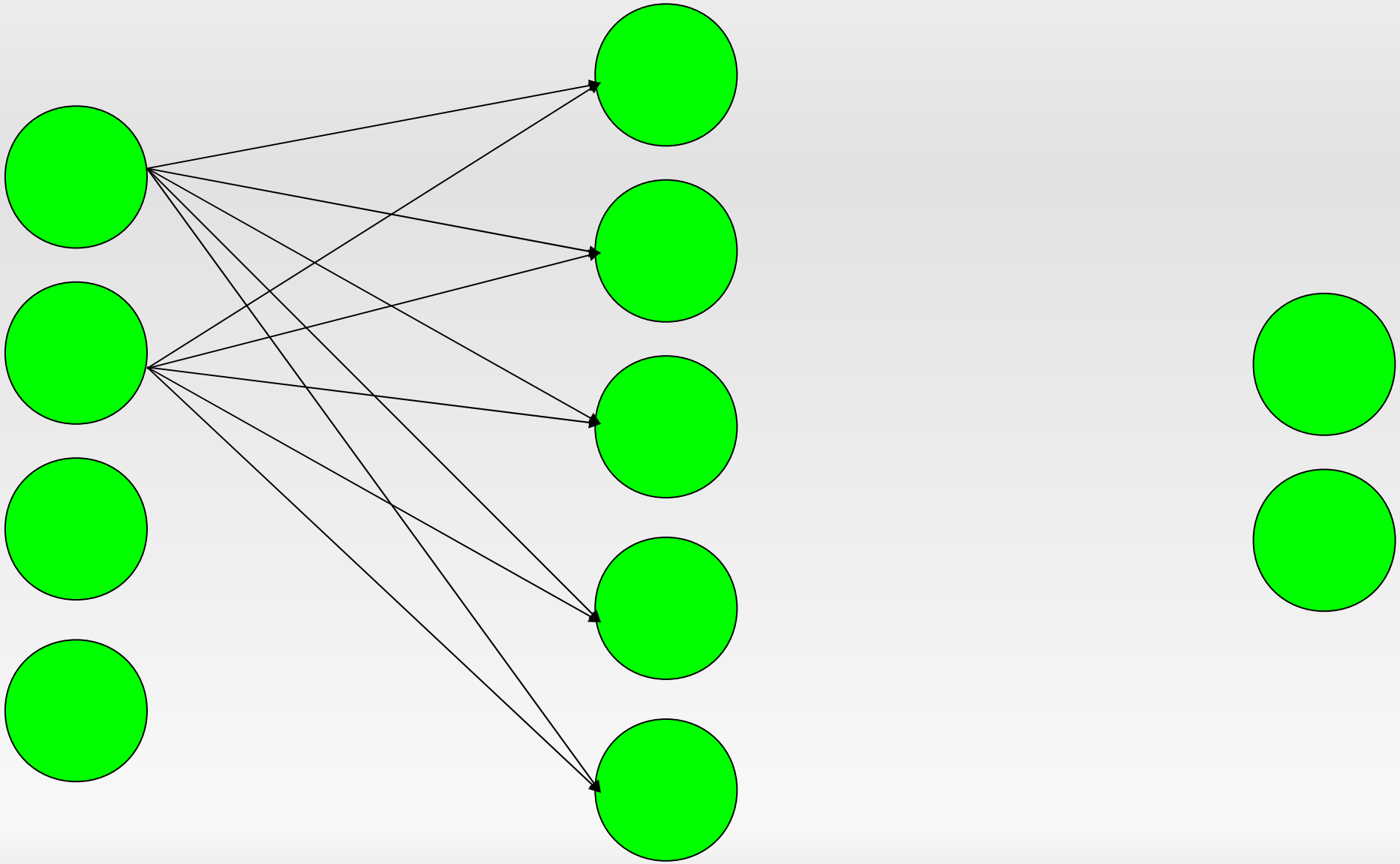
La rete *feed-forward*



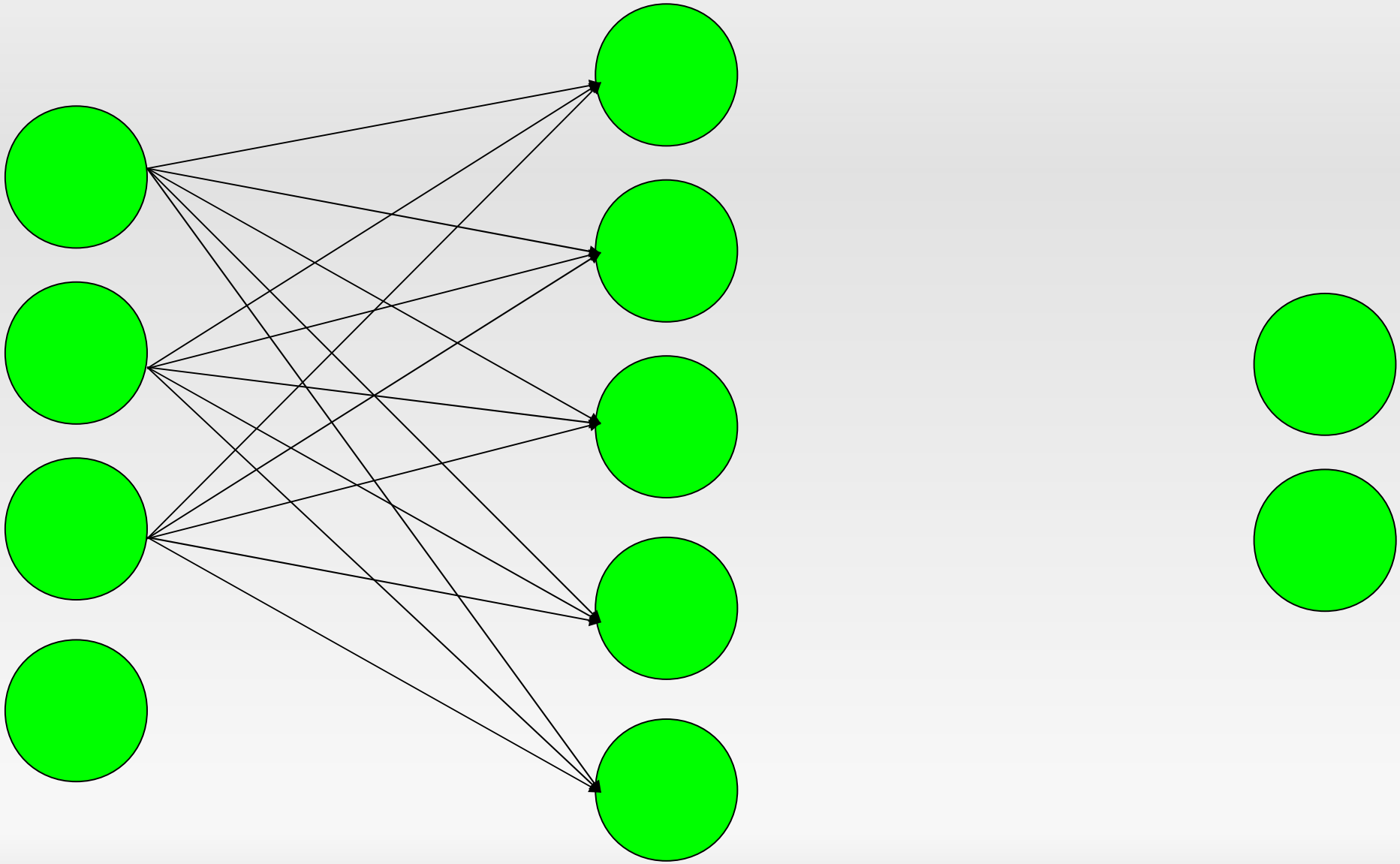
La rete *feed-forward*



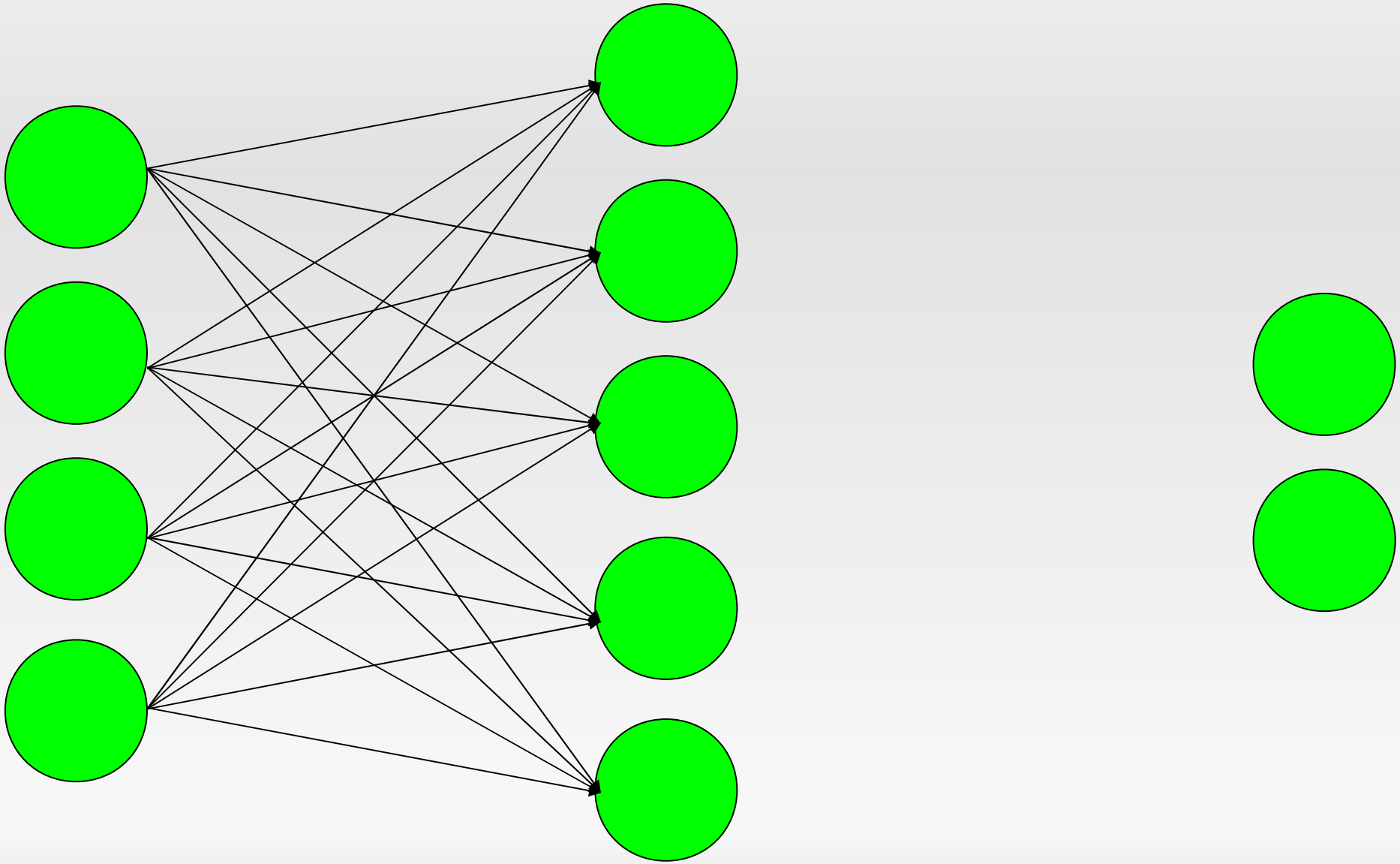
La rete *feed-forward*



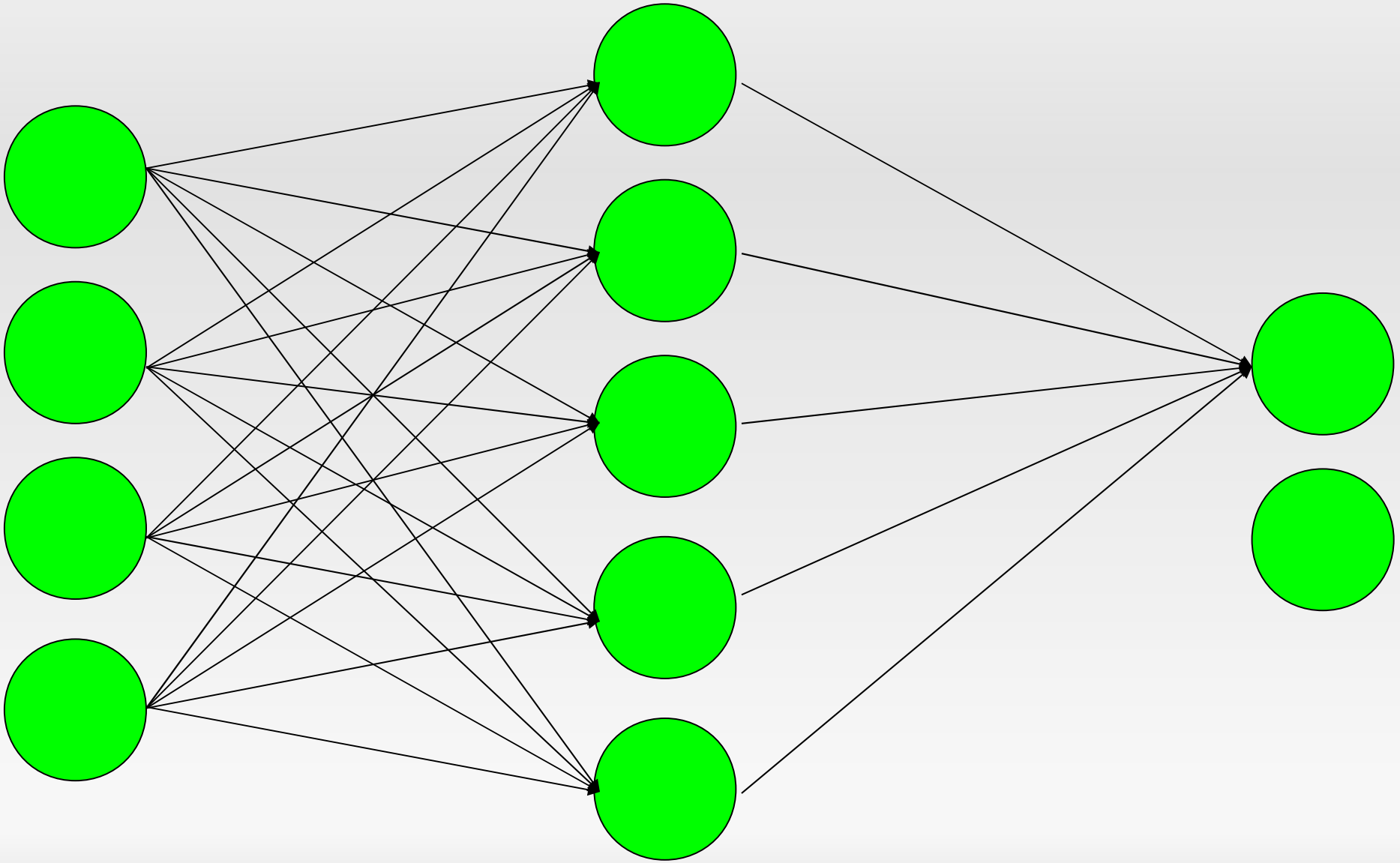
La rete *feed-forward*



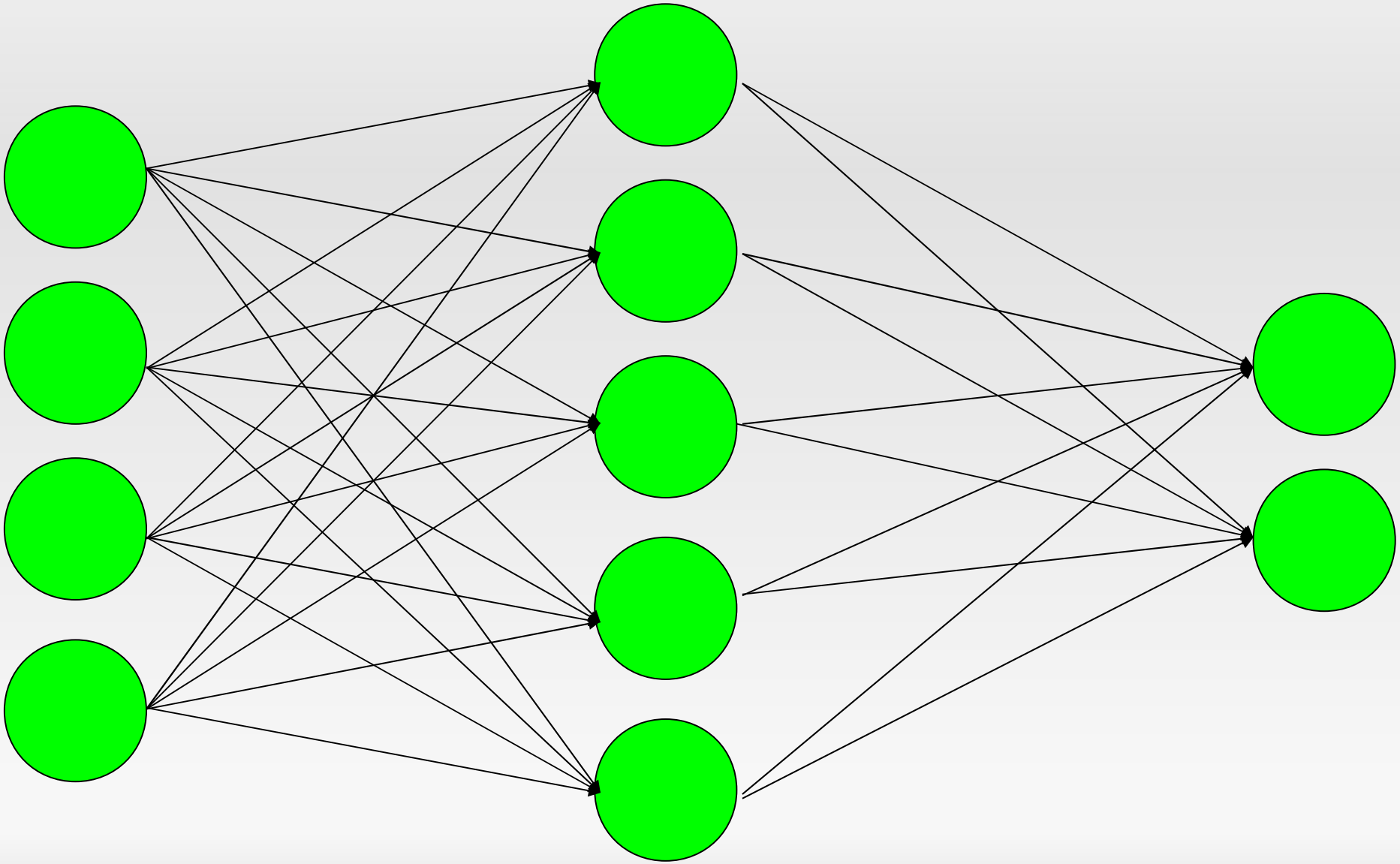
La rete *feed-forward*



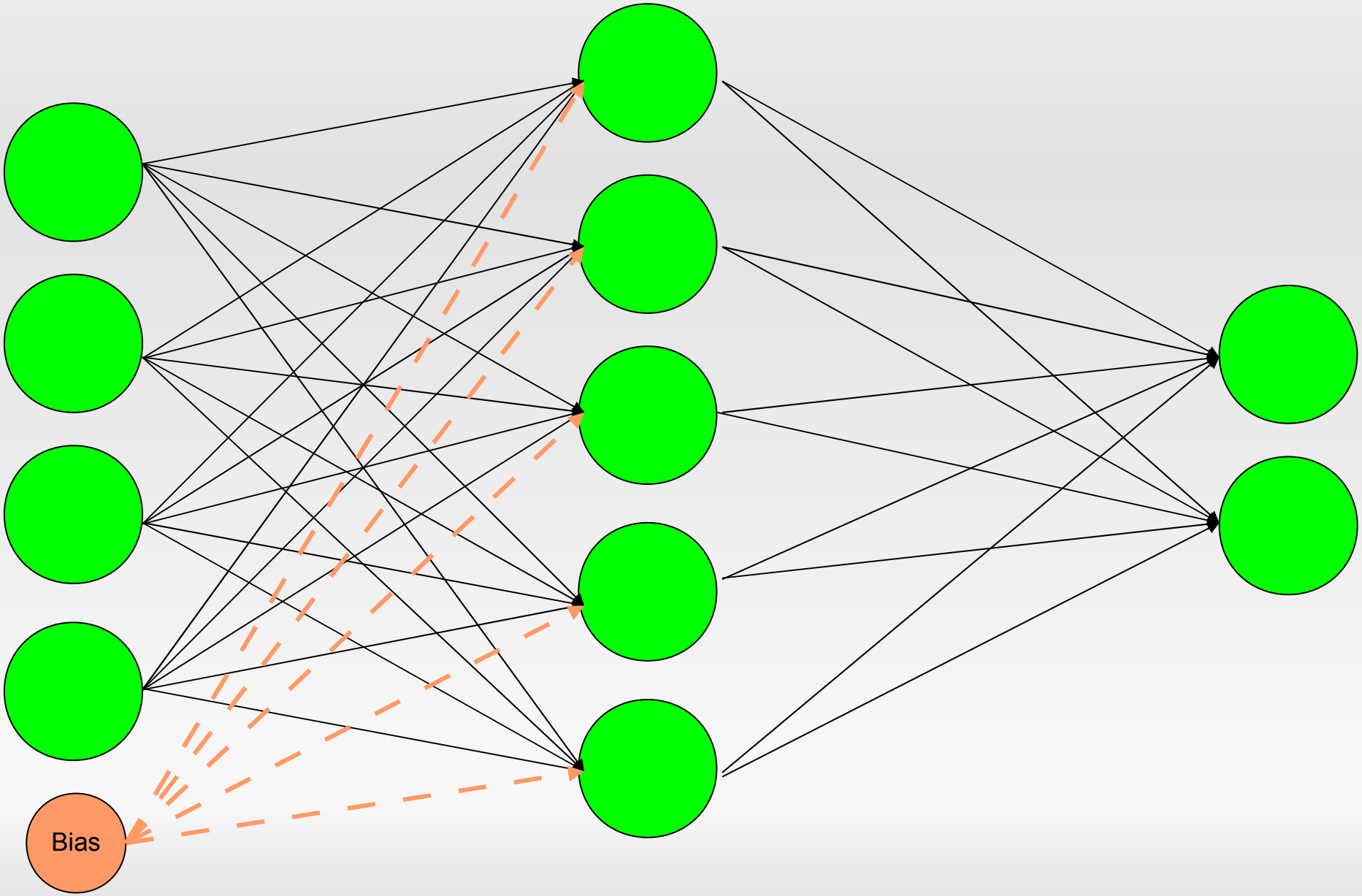
La rete *feed-forward*



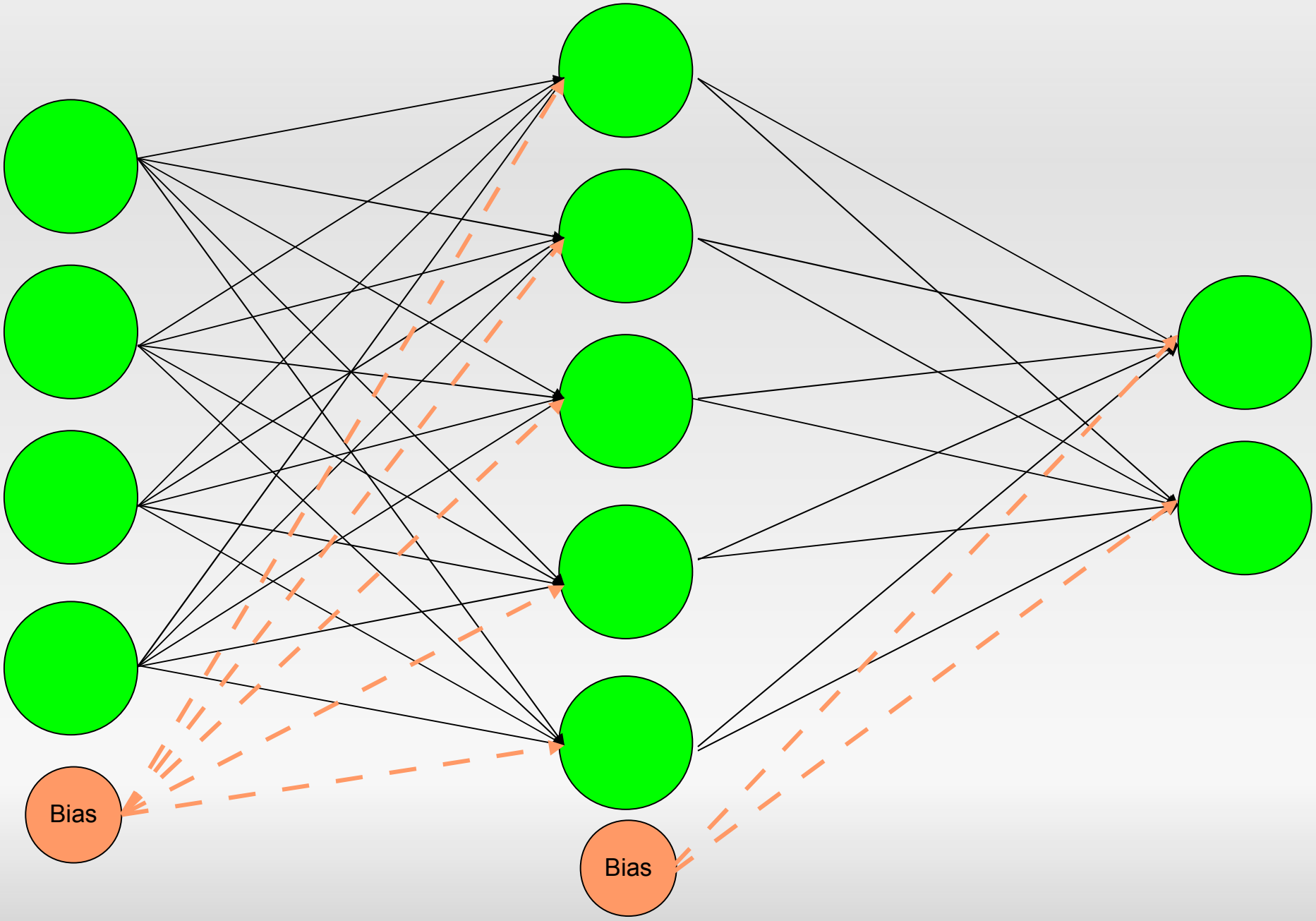
La rete *feed-forward*



La rete *feed-forward* + *BIAS*

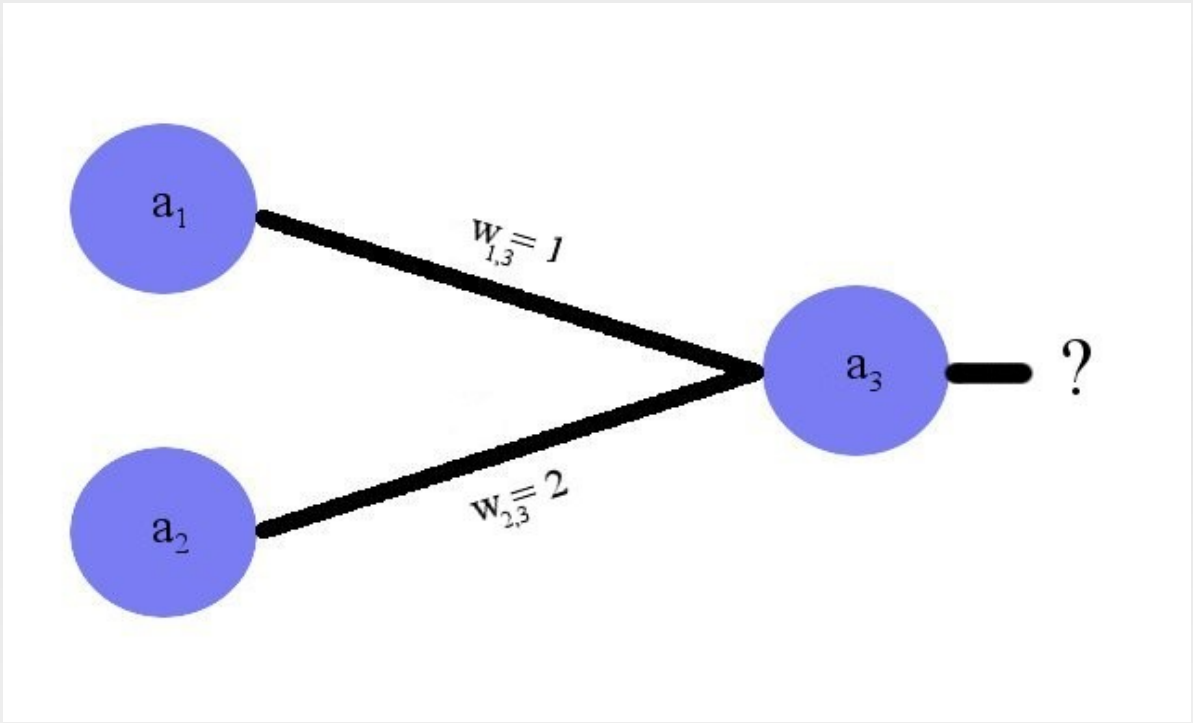


La rete *feed-forward* + *BIAS*



Esempio di calcolo del valore di attivazione

$$a_1 = 1 \qquad w_{1,3} = 1$$
$$a_2 = 0,5 \qquad w_{2,3} = 2$$

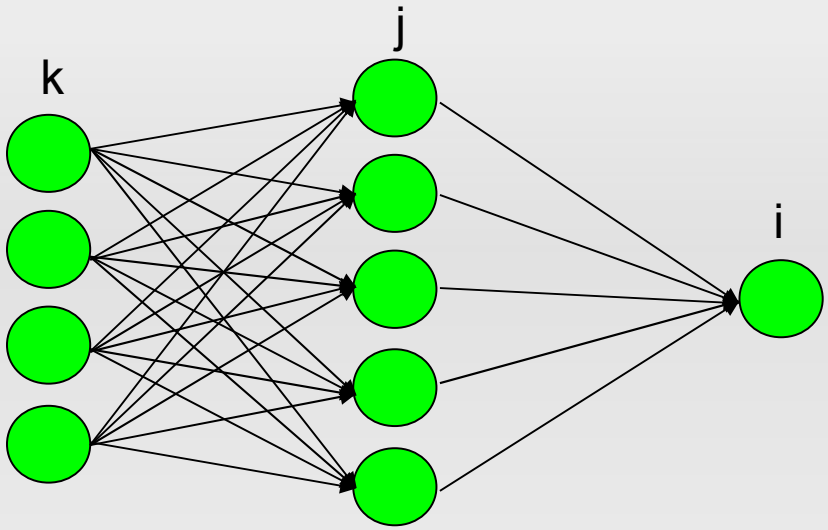


$$a_3 = f\left[(a_1 \cdot w_{1,3}) + (a_2 \cdot w_{2,3}) \right]$$

$$a_3 = f\left[(1 \cdot 1) + (0,5 \cdot 2) \right] = f(2)$$

$$a_3 = \frac{1}{1 + e^{-2}} \simeq 0,88$$

L'algoritmo di retro-propagazione dell'errore



$$w_{i,j} = w_{i,j} + \varepsilon \cdot \Delta_i$$

$$\Delta_i = Err_i \cdot f'\left(\sum_j w_{i,j} \cdot a_i\right)$$

$$Err_i = \left(\text{valore desiderato}_i - \text{valore ottenuto}_i \right)$$

$$f'(x) = \frac{e^x}{e^{-2x} + 2e^{-x} + 1}$$

$$f'(x) = y - y^2$$

L'algoritmo di retro-propagazione dell'errore

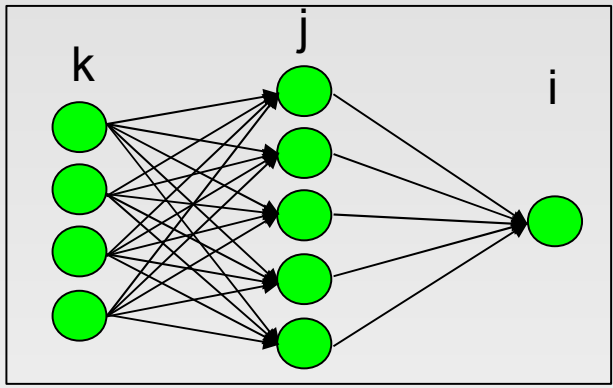
Delta dei neuroni dello strato intermedio

$$\Delta_j = Err_j \cdot f'\left(\sum_k w_{k,j} \cdot a_j\right)$$

$$Err_j = \sum_i w_{i,j} \cdot \Delta_i$$

Pesi sinaptici tra strato d'ingresso e intermedio

$$w_{k,j} = w_{k,j} + \varepsilon \cdot \Delta_j$$

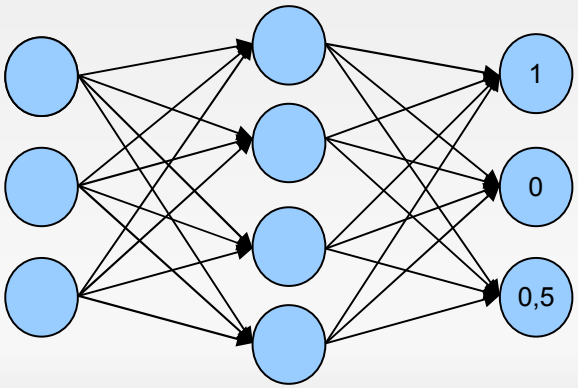


Errore quadratico medio

Errore totale della rete neurale:

$$E_{tot} = \frac{1}{2} \cdot \sum_i \left(D_i - Y_i \right)^2$$

Esempio:



Valore ottenuto: 1 ; valore desiderato: 0

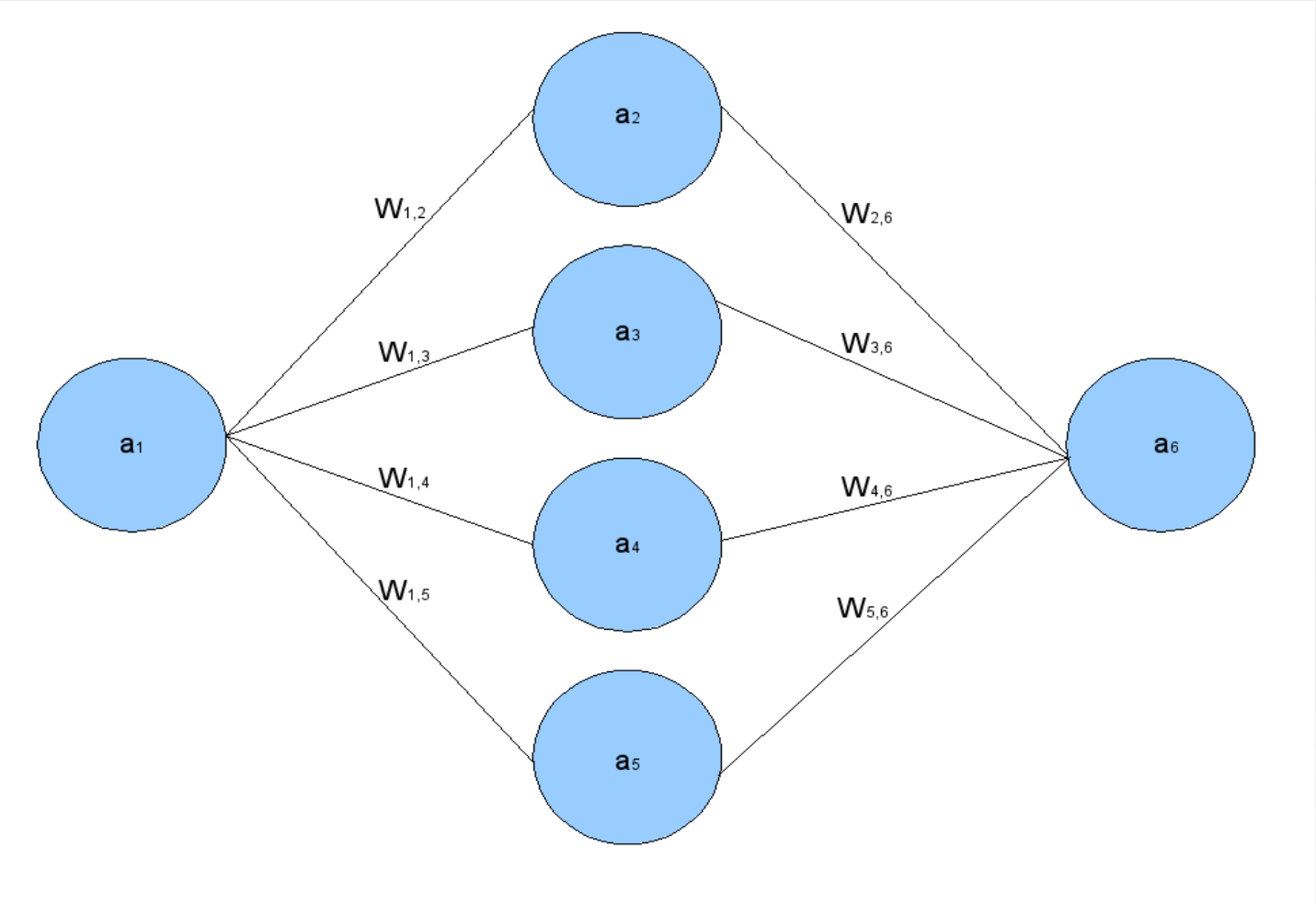
Valore ottenuto: 0 ; valore desiderato: 0,3

Valore ottenuto: 0,5 ; valore desiderato: 1

$$E_{tot} = \frac{1}{2} \cdot \left[(0-1)^2 + (0{,}3-0)^2 + (1-0{,}5)^2 \right]$$

$$E_{tot} \simeq 0{,}67$$

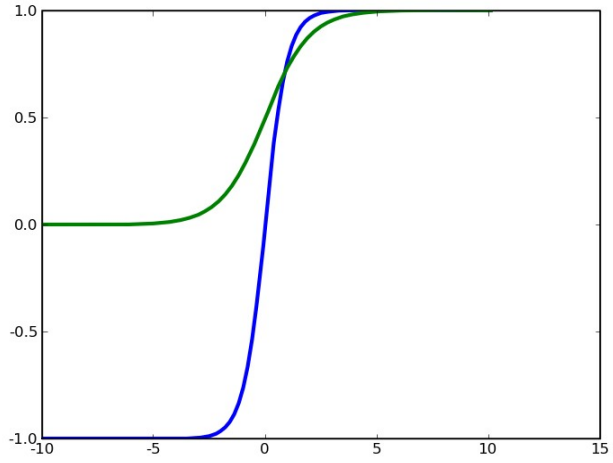
Applicazione: approssimazione della funzione seno



Angolo	-180°	-120°	-90°	-60°	0°	60°	90°	120°	180°
	↓	↓	↓	↓	↓	↓	↓	↓	↓
Valore di seno	0	$\frac{-\sqrt{3}}{2}$	-1	$\frac{-\sqrt{3}}{2}$	0	$\frac{\sqrt{3}}{2}$	1	$\frac{\sqrt{3}}{2}$	0

Applicazione: approssimazione della funzione seno

- La tangente iperbolica
  - Dominio: tutti i numeri reali
  - Codominio: $(-1\,;1)$



- Set di esempi:

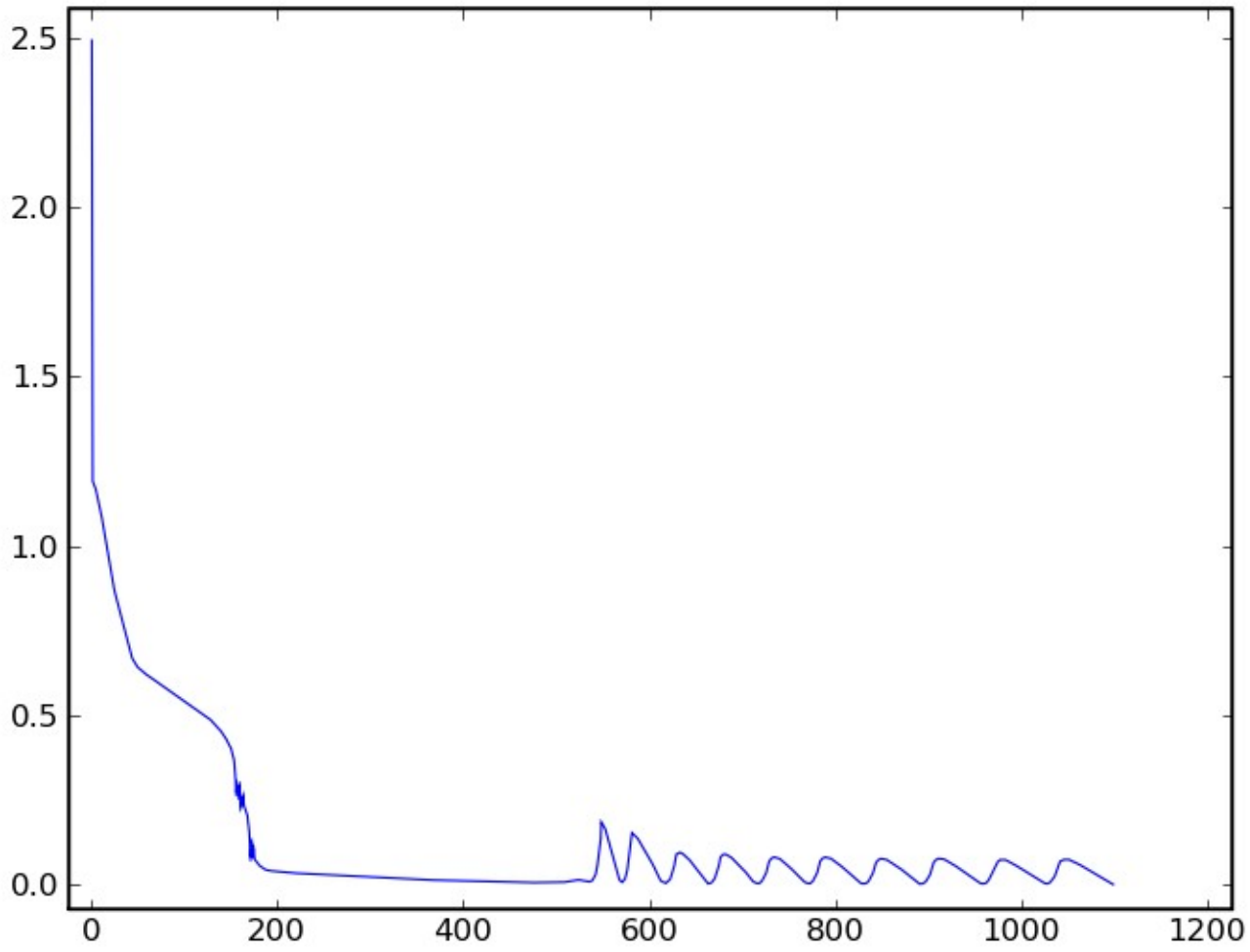
$$Per\ a_1\ uguale\ a\ \frac{-180}{180} = -1 \quad \rightarrow\ a_6\ dovrà\ risultare\ uguale\ a\ 0$$

$$Per\ a_1\ uguale\ a\ \frac{-120}{180} = -0{,}66 \quad \rightarrow\ a_6\ dovrà\ risultare\ uguale\ a\ \frac{-\sqrt{3}}{2}$$

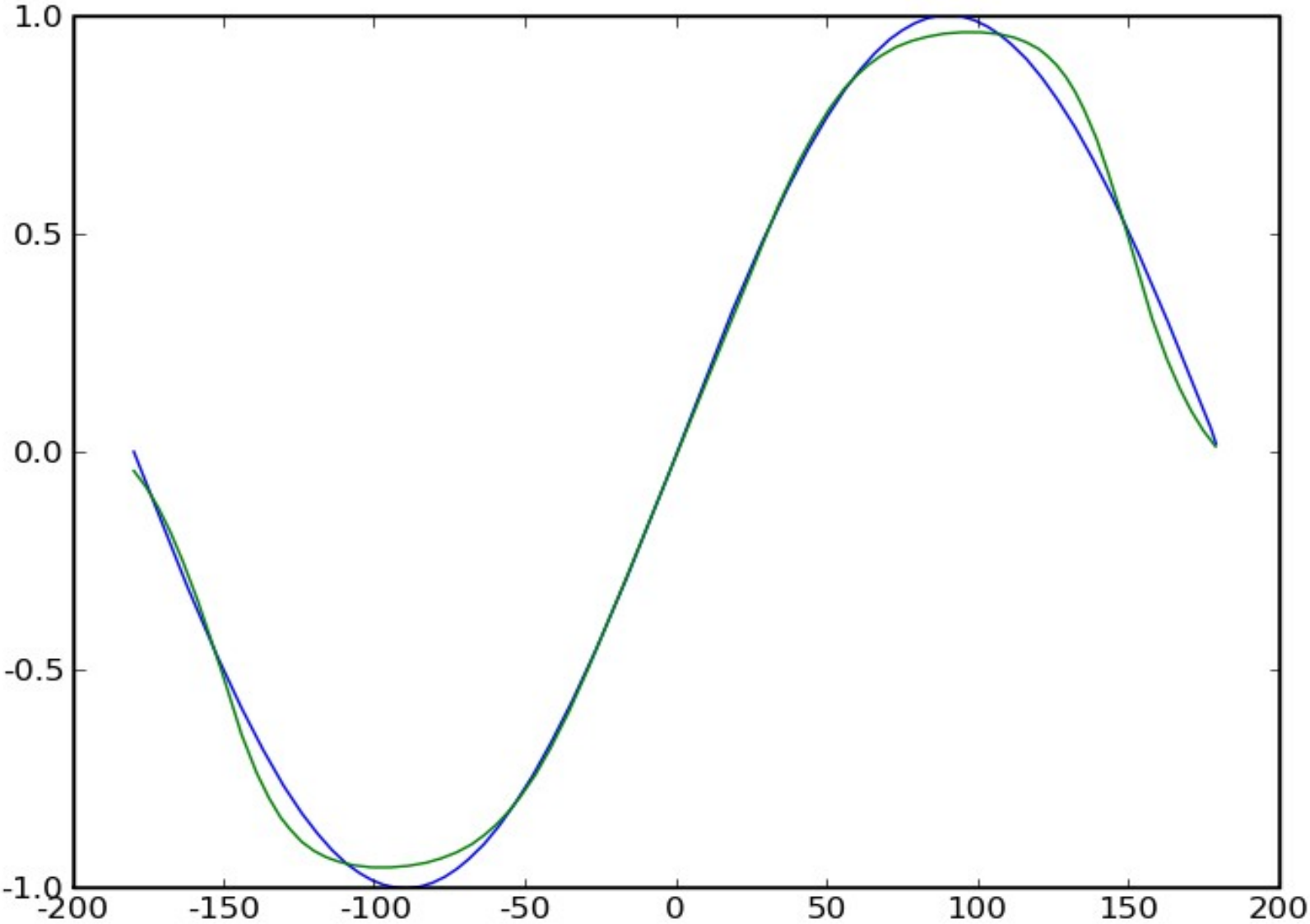
$$Per\ a_1\ uguale\ a\ \frac{-90}{180} = -0{,}5 \quad \rightarrow\ a_6\ dovrà\ risultare\ uguale\ a\ -1$$

...

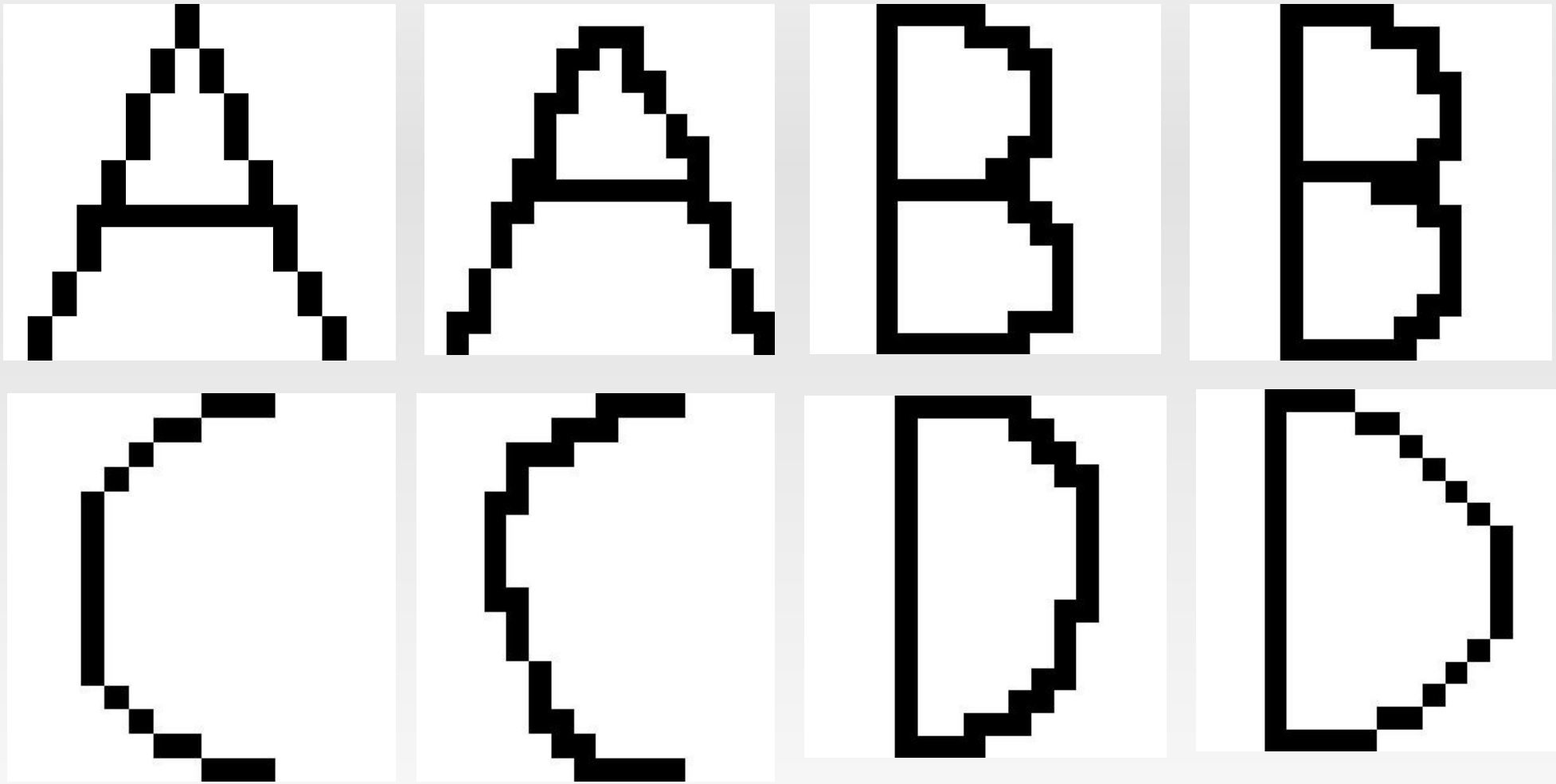
Risultati dell'applicazione



Risultati dell'applicazione



Sistema OCR rudimentale



Sistema OCR rudimentale

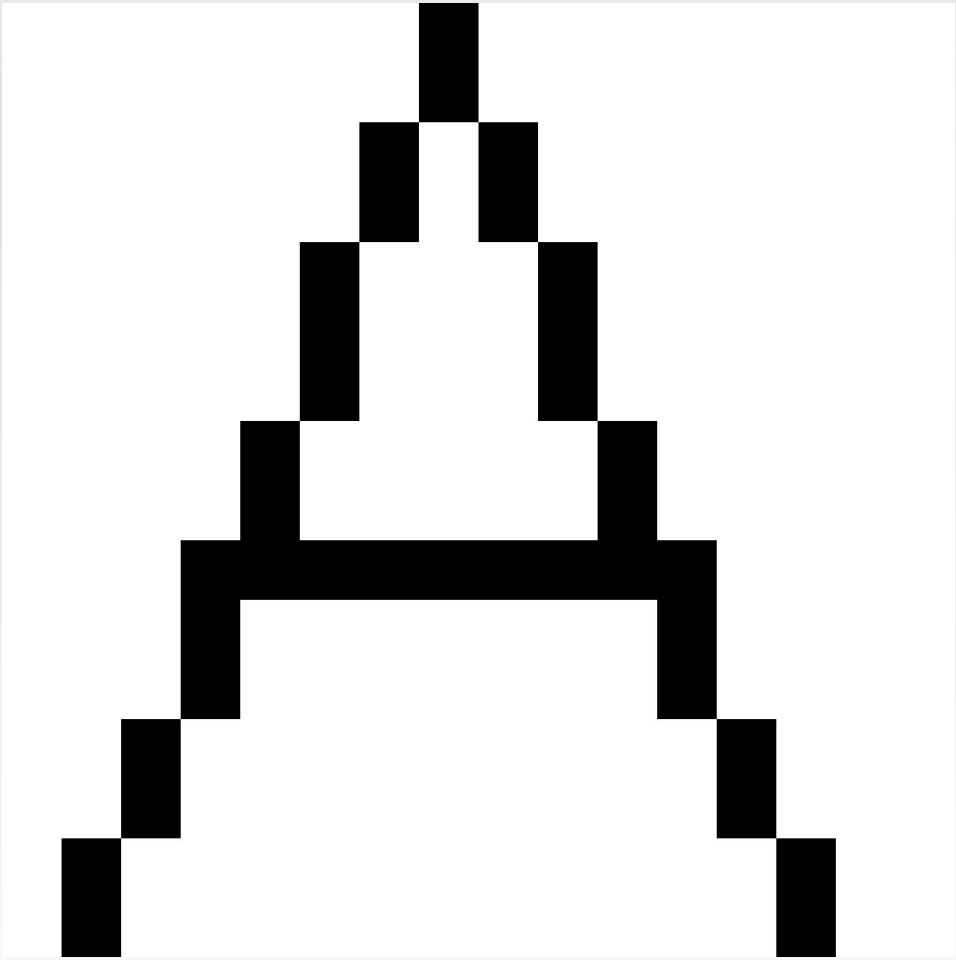
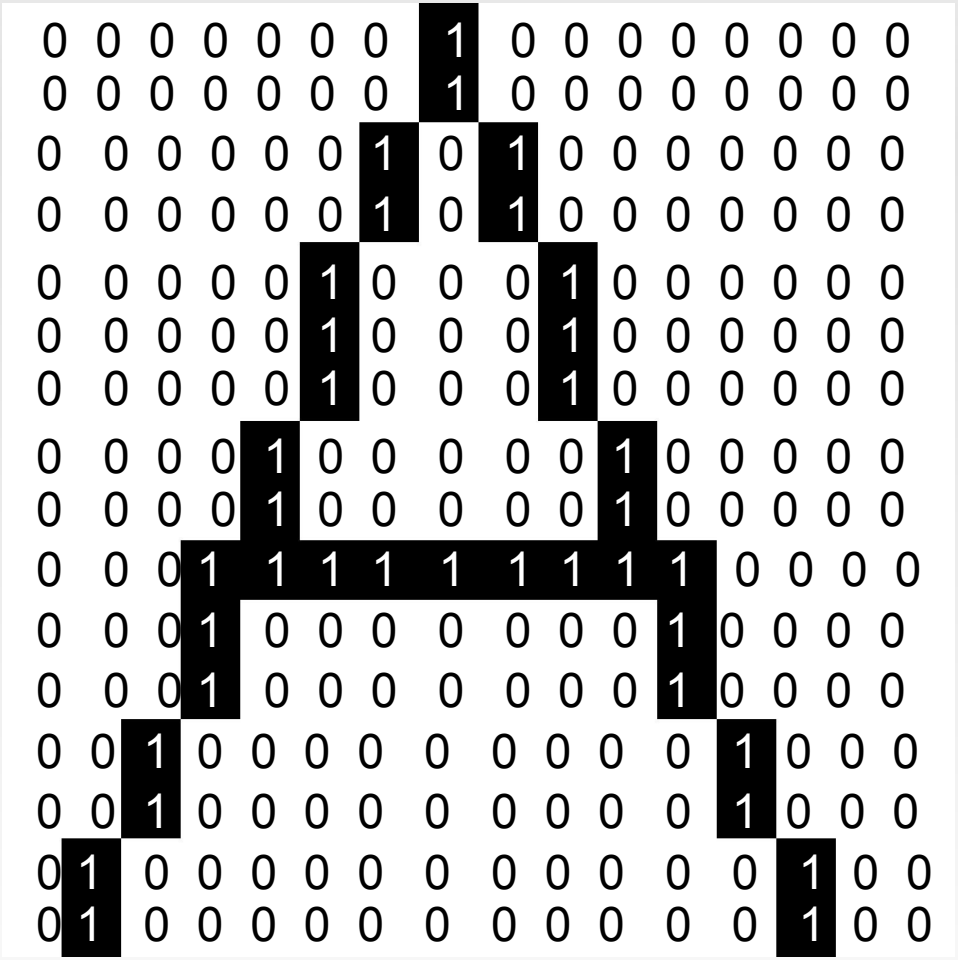


Immagine binaria



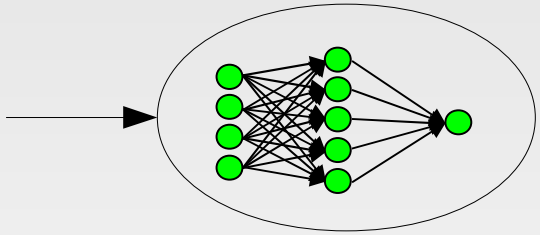
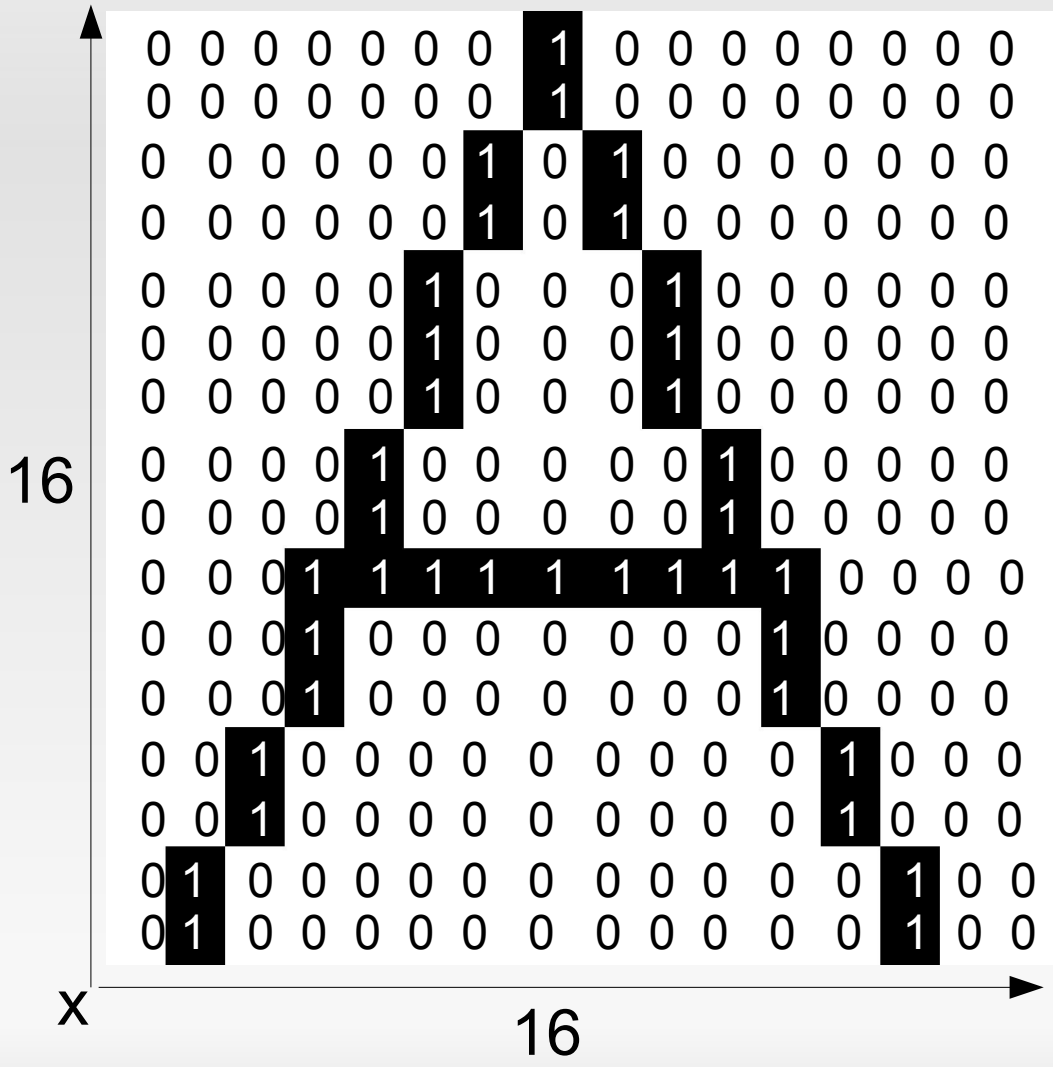


Immagine binaria



= 256 bit, cioè 256 ingressi

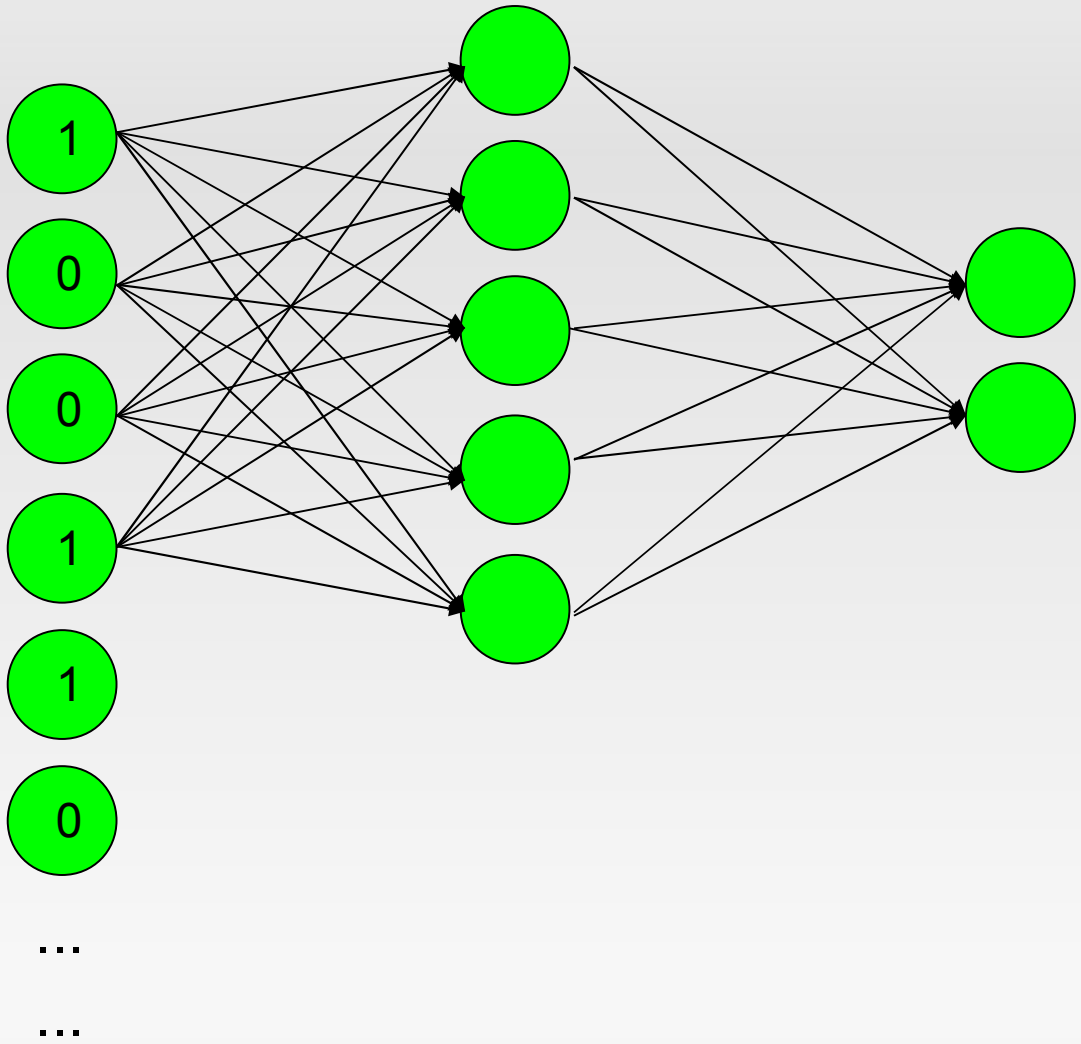
Struttura della rete

- 256 neuroni in ingresso (16*16)
- 16 neuroni in strato intermedio
- 4 neuroni in uscita (uno per lettera)

- Errore quadratico medio abbassato fino a 0.002314
- La rete ha compreso gli 8 esempi ed è riuscita a riconoscere le altre 40 immagini di test senza errori

Immagine binaria

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



E quanti neuroni hidden metto?

- Il numero di neuroni nello strato nascosto in una rete neurale deve dipendere dal numero di fattori o ”features” di cui si vuole tenere conto.
- Purtroppo raramente è facile capire quante sono effettivamente queste ”features”
- Esistono alcune formule, anche se restano poco affidabili e solo come relazioni ”empiriche”

$$N_n = \frac{\text{Numero Esempi} \cdot \text{Errore percentuale consentito}}{\text{Numero input} + \text{Numero output}}$$

$$\text{Numero esempi} > 10 \cdot \text{Numero pesi}$$

Pregi delle reti neurali

- Applicazioni in contesti con dati soggetti a rumore o parzialmente errati
- Possono essere utilizzate in contesti di riconoscimento di immagini o di suoni in modo relativamente semplice e veloce.
- Utilissime in campo di previsioni: sia finanziarie (come ad esempio previsioni di bancarotta, previsione del prezzo di azioni) che meteorologiche o in altri settori.
- Hanno il pregio assoluto di poter astrarre a delle relazioni tra i dati in modo “indipendente” e capire i “motivi” che determinano i risultati

Difetti delle reti neurali

- I risultati molto efficienti ma non spiegabili in modo chiaro: dobbiamo accettarli “così come sono”. Un po' come una “scatola nera”.
- Necessitano spesso set di esempi molto ampi
- Per alcune applicazioni complesse l'addestramento può richiedere relativamente parecchio tempo.
- Non esiste la “rete perfetta”, ma la realizzazione di una struttura adatta al problema da risolvere dipende molto al creatore.

Bpnn.py

- È uno script/lib python comodo per fare qualche test
- Reperibile a http://arctrix.com/nas/python/bpnn.py
- Simula una rete feed-forward a tre strati, utilizzando l'algoritmo back-propagation

Bpnn.py – Esempio applicativo

```
import bpnn
net = bpnn.NN(2, 2, 1)
pat = [
        [[0,0], [0]],
        [[0,1], [1]],
        [[1,0], [1]],
        [[1,1], [0]]
    ]
net.train(pat)
net.test(pat)
```

```
otacon22@desktop:~$ python bpnn.py

error 0.942497
error 0.042867
error 0.003480
error 0.001642
error 0.001062
error 0.000782
error 0.000794
error 0.000527
error 0.000442
error 0.000380
[0, 0] -> [0.02560857904121879 5]
[0, 1] -> [0.9818457844779476 8]
[1, 0] -> [0.9817074256406621 6]
[1, 1] -> [-0.02103006443981345 1]
```

Fast Artificial Neural Network Library (FANN)

- Reperibile a http://leenissen.dk/fann/
- Vantaggioso perchè ha il bindings su diversi linguaggi:
  - .NET, C++, Java, Ada, PERL, PHP, Python, Ruby, Delphi, TCL, Lua, Haskel, Mathematica, Matlab, Prolog, Octave, Pure Data.
- È dotata anche di vari tool grafici
- Permette molte operazioni avanzate, tra cui ad esempio settare funzioni di trasferimento diverse per strati differenti

Fann example

```
#!/usr/bin/python
from pyfann import fann

connection_rate = 1
learning_rate = 0.7
num_input = 2
num_neurons_hidden = 4
num_output = 1

desired_error = 0.0001
max_iterations = 100000
iterations_between_reports = 1000

ann = fann.create(connection_rate, (num_input, num_neurons_hidden, num_output))
ann.set_learning_rate(learning_rate)
ann.set_activation_function_output(fann.SIGMOID_SYMMETRIC_STEPWISE)

ann.train_on_file("../../examples/xor.data", max_iterations, iterations_between_reports,
desired_error)

ann.save("nets/xor_float.net")
```