■ Esempi di applicazioni di reti neurali

Ogni problema di rete neurale segue la stessa procedura di base: (1) caricare il set di dati, (2) inizializzare la rete, (3) addestrare la rete (training), e (4) validare il modello. Qui i passaggi fondamentali comuni a tutti i problemi delle reti neurali sono chiariti con due esempi: un problema di classificazione e un problema di approssimazione di funzioni.

■ Problema di classificazione

Questa sezione descrive un semplice problema di classificazione. Questo esempio bidimensionale è istruttivo perché i dati e il classificatore possono essere visualizzati molto facilmente. I dati utilizzati in questo esempio sono memorizzati nell'array **x** per l'input e **y** per l'output. Per comprendere il problema, si supponga che i dati di input rappresentino i parametri di patologia per 40 diversi soggetti e i dati di uscita rappresentino la diagnosi per ciascun soggetto (1=sano, 0=malato). Ci sono due classi possibili in questo esempio.

Carichiamo i dati da elaborare.

```
<<NeuralNetworks`
```

```
<< twoclasses.dat;
```

4.82138	3.83606
6.34574	4.29422
5.95773	2.23357
4.6027	4.24406
5.6715	6.53738
6.04556	6.20115
5.7479	4.47444
3.96408	6.9519
4.22316	6.39823
4.11563	5.2621
7.14588	7.23001
5.62569	4.09736
3.30412	6.19516
7.51823	3.68374
2.92871	3.50996
4.99486	7.27725
3.91102	5.64594
5.86308	5.75472
4.20995	4.93891
5.45854	3.04683
0.643767	1.40386
−0.516567	0.904904
1.73589	0.60667
−0.634848	0.0556446
−0.38834	−0.543183
−0.857059	−0.356523
1.01237	0.342814
−0.565642	1.36846
−0.351247	−0.409551
0.3309	0.977012
0.0723711	0.0722728
−1.57544	0.981475
0.805269	−0.299545
0.893102	0.482046
0.897225	−0.744046
−0.586793	−0.386432
0.0596479	0.830947
1.9795	0.376903
0.766149	1.82142
−0.136265	−0.659328

Poiché ci sono due sole classi possibili, l'ouput può essere memorizzato in una colonna, con un 1 o 0 indicante la classe cui il soggetto appartiene.

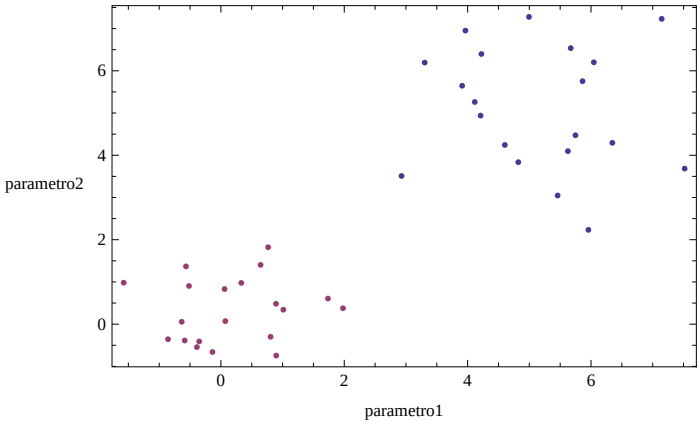
Esaminiamo i dati di output.

```
y // TraditionalForm
```

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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}

Rappresentiamo graficamente i dati (gli assi rappresentano i due parametri dei soggetti):

```
NetClassificationPlot[x,y,FrameLabel→{"parametro1","parametro2"},RotateLabel→False]
```



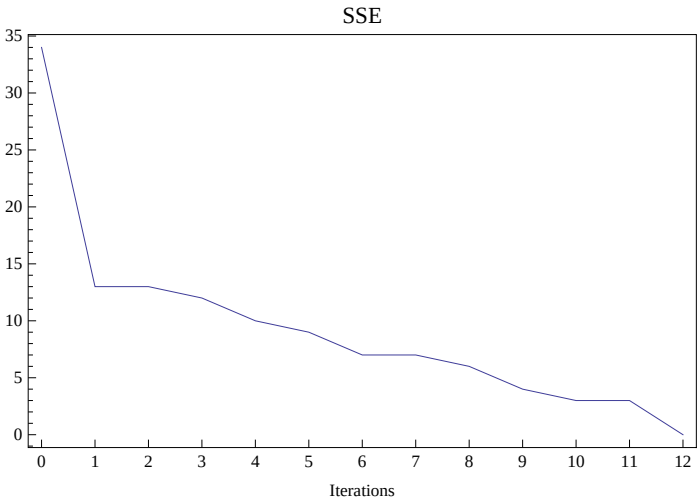
Il grafico mostra chiaramente che i dati vengono suddivisi in due classi e che dovrebbe essere possibile dividere i gruppi con una linea. La retta viene determinata durante il processo di addestramento (training), e la rete neurale per la classificazione addestrata con successo restituirà la classe corretta cui il soggetto appartiene, dati i suoi parametri.

Una misura dell'adattamento (o indice delle prestazioni) che l'algoritmo di addestramento deve minimizzare deve essere scelta prima di procedere all'addestramento. Per i problemi di classificazione, il criterio è impostato al numero di campioni di dati erroneamente classificati. Il classificatore ha classificato correttamente tutti i dati quando il criterio è pari a zero.

Un Perceptron è il più semplice tipo di rete neurale per la classificazione, e verrà utilizzato per illustrare l'addestramento in questo esempio.

Inizializziamo e addestriamo un classificatore di tipo Perceptron sui dati acquisiti in precedenza.

```
{per, fitrecord} = PerceptronFit[x, y];
```



Si noti che di solito si ottengono risultati leggermente diversi se si ripete il training. Ciò è dovuto alla inizializzazione casuale del Perceptron. Come risultato di questo, i pesi parametrici del Perceptron saranno anche diversi per ogni valutazione e si otterranno perciò classificatori diversi.

Al termine del training, una sintesi del processo di addestramento viene mostrata nel grafico dell'errore quadratico totale (SSE, summed squared error) rispetto al numero di iterazioni. Il precedente grafico è la sintesi del processo di addestramento per questo esempio. Si può vedere che l'SSE tende a zero man mano che aumenta il numero di iterazioni.

Dopo aver effettuato on successo l'addestramento del Perceptron (**per**), possiamo utilizzarlo per classificare nuovi dati di input.

Classifichiamo un nuovo soggetto con parametro1=6 e parametro2=7:

```
per[{6., 7.}]
```

{1}

Il soggetto viene classificato come sano.

Esaminiamo in dettaglio i pesi e la struttura del classificatore.

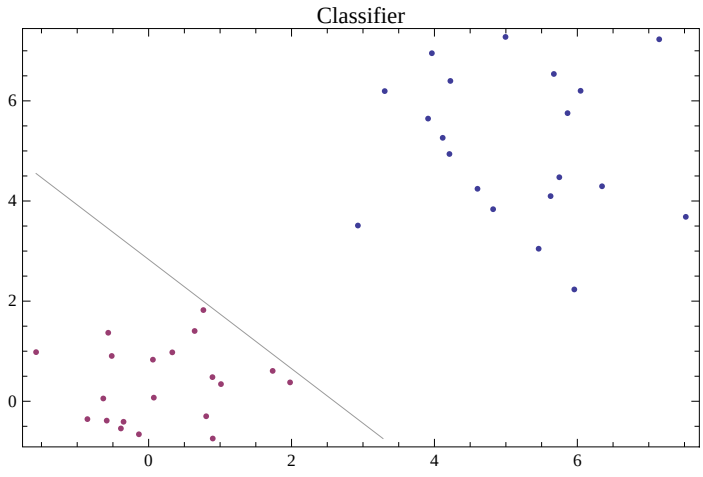
```
per[{a, b}]
```

$\{\theta(70.6236\, a + 64.775\, b - 183.427)\}$

Si osservi che i valori numerici dei pesi possono variare al ripetersi dell'esempio.

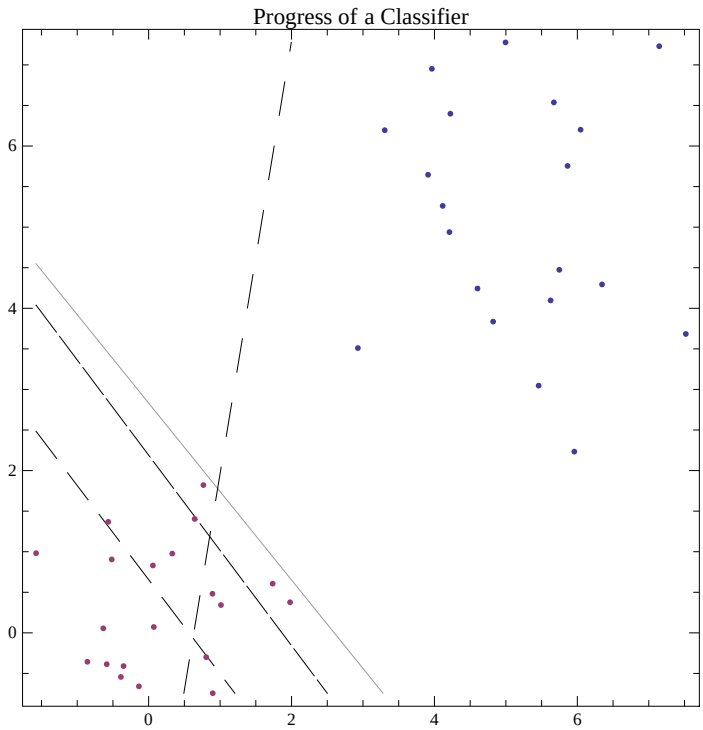
Possiamo illustrare la rete addestrata in vari modi. Il classificatore addestrato può, ad esempio, essere visualizzato insieme ai dati. Questo tipo di grafico è illustrato utilizzando i risultati del problema di classificazione bidimensionale. Per questo esempio, un classificatore corretto divide le due classi con una linea. L'esatta posizione di tale linea dipende da quale particolare soluzione è stata trovata nel training.

```
NetPlot[per, x, y]
```



Illustriamo il processo di addestramento del Perceptron.

```
NetPlot[fitrecord, x, y]
```



Il grafico mostra la classificazione iniziale ed il suo miglioramento durante il processo di addestramento.

■ Approssimazione di una funzione

Questa sezione contiene un problema di approssimazione unidimensionale risolto con una rete FeedForward. Problemi con dimensioni superiori, tranne per i grafici dei dati, possono essere gestiti in modo simile.

Carichiamo i dati da elaborare.

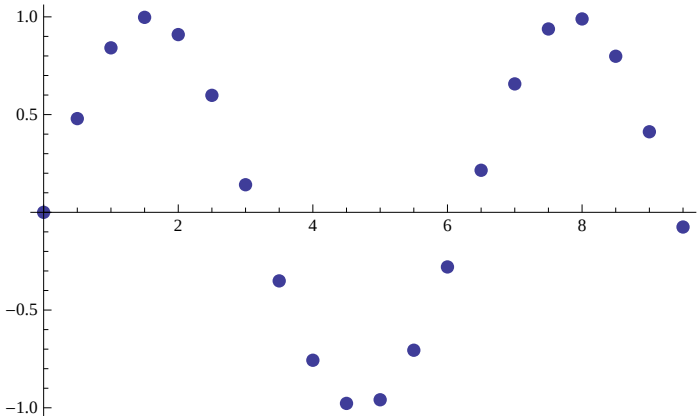
```
<<onedimfunc.dat;
```

x	y
0.	0.
0.5	0.479426
1.	0.841471
1.5	0.997495
2.	0.909297
2.5	0.598472
3.	0.14112
3.5	−0.350783
4.	−0.756802
4.5	−0.97753
5.	−0.958924
5.5	−0.70554
6.	−0.279415
6.5	0.21512
7.	0.656987
7.5	0.938
8.	0.989358
8.5	0.798487
9.	0.412118
9.5	−0.0751511

I dati di input e output sono memorizzati negli array **x** e **y**, rispettivamente. Si presume che le coppie ingresso-uscita siano collegate dalla relazione $y=f(x)$, dove f è una funzione non specificata. I dati saranno utilizzati per istruire una rete FeedForward che sarà un'approssimazione della reale funzione f.

Per giustificare l'uso di una rete neurale, immaginiamo che sia x che y siano valori misurati di alcuni prodotti in una fabbrica, ma che y possa essere misurato solo distruggendo il prodotto. Diversi campioni del prodotto sono stati distrutti per ottenere il set di dati. Se un modello di rete neurale può trovare un rapporto tra x ed y in base a questi dati, allora i valori futuri di y potrebbero essere calcolati da una misurazione di x senza distruggere il prodotto.

Rappresentiamo graficamente i dati.



Questo è un esempio molto banale: i dati sono stati generati con una sinusoide. Una rete FeedForward sarà addestrata ad approssimare la funzione.

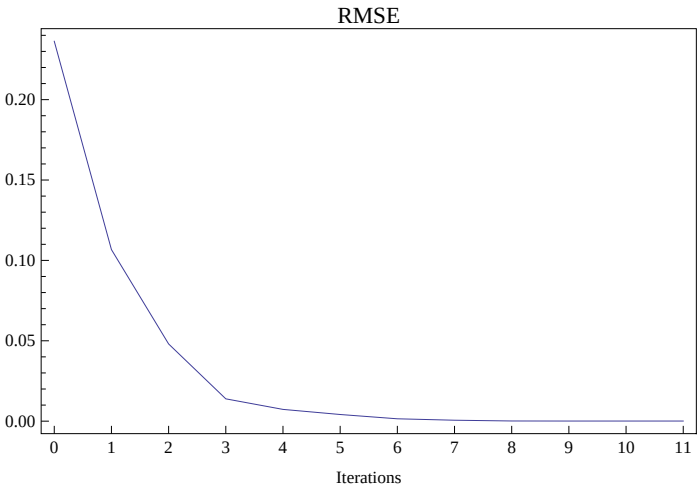
Inizializziamo una rete FeedForward con quattro neuroni.

```
fdfrwrd = InitializeFeedForwardNet[x, y, {4}]
```

```
FeedForwardNet[{{w1, w2}}, {Neuron → Sigmoid, FixedParameters → None, AccumulatedIterations → 0,
    CreationDate → {2013, 3, 30, 15, 10, 48.041837}, OutputNonlinearity → None, NumberOfInputs → 1}]
```

Addestriamo la rete inizializzata.

```
{fdfrwrd2, fitrecord} = NeuralFit[fdfrwrd, x, y];
```



Si noti che di solito si ottengono risultati leggermente diversi se si ripete l'inizializzazione e il comando di training. Ciò è dovuto all'inizializzazione parzialmente casuale della rete FeedForward.

Come nell'esempio precedente, al termine dell'addestramento il miglioramento del *fit* è riassunto in un grafico dell'RMSE (Root Mean Squared Error) nella previsione della rete neurale rispetto alle iterazioni. Al termine dell'addestramento, si può ricevere un avviso che il training non converga. Di solito è meglio di esaminare

visivamente la diminuzione dell'RMSE nel grafico per decidere se sono necessarie ulteriori iterazioni di addestramento. Si assume qui che la rete sia stata correttamente addestrata, anche se in seguito rappresenteremo graficamente il modello per confrontarlo con i dati.

Il primo argomento di outout del comando **NeuralFit** è la rete FeedForward addestrata. Il secondo argomento è un training record contenente le informazioni sulla procedura di addestramento.

La rete neurale addestrata può ora essere applicata ad un valore *x* per stimare *y = f(x)*.

Facciamo una stima di y per x=3

```
fdfrwrd2[{3}]
```

{0.141201}

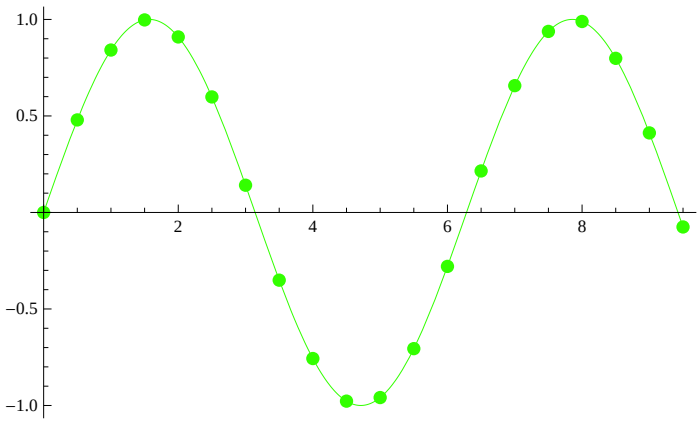
Esaminiamo la struttura ed i parametri della rete feedforward.

```
fdfrwrd2[{a}]
```

$$\left\{\frac{5.82366}{e^{0.33647-0.990612\,a}+1}-\frac{8.76232}{e^{2.66706-0.823978\,a}+1}-\frac{5.71852}{e^{9.09105-1.00045\,a}+1}+6.77881-\frac{8.68761}{e^{0.827466\,a-5.11268}+1}\right\}$$

Tracciamo il grafico della stima della funzione insieme ai dati.

```
NetPlot[fdfrwrd2, x, y, DataFormat → FunctionPlot,
 PlotStyle → {Hue[0.3], PointSize[0.02]}]
```



Come si può vedere dal grafico, l'approssimazione è perfetta!