Riconoscimento di cifre manoscritte tramite una rete neurale

Andrea Burattin

3 luglio 2008

Argomenti discussi

Il problema

Il problema

Il problema

Molte applicazioni pratiche. Alcuni esempi:

- Sistemi di smistamento automatico della posta cartacea, basato sul riconoscimento del CAP scritto nelle buste delle lettere
- Inserimento automatico degli importi delle tasse letti dai bollettini
- Riconoscimento automatico degli input per computer palmari

Il training set

Il training set

Il training set

Per la fase di training si è usato "*The MNIST database of handwritten digits*":

- 60 000 esempi di trainig
- 10 000 esempi per il validation set

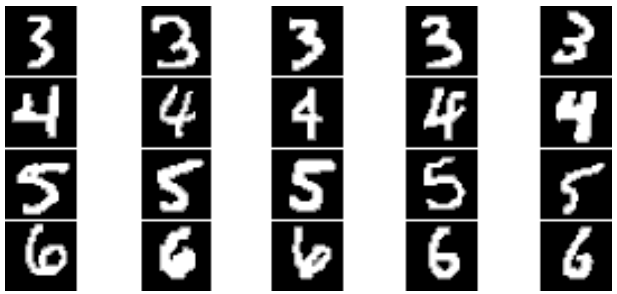


Figura: Esempio di dati estratti dal database MNIST.

Codifica dei file

Struttura degli esempi:

1. Immagini originali, del NIST, in bianco e nero
2. Normalizzate a 20×20 pixel, in scala di grigi (per l'*antialiasing*)
3. Centrate nel *centro di massa dei pixel* in un'area di 28×28

Codifica dei file

Struttura degli esempi:

1. Immagini originali, del NIST, in bianco e nero
2. Normalizzate a 20×20 pixel, in scala di grigi (per l'*antialiasing*)
3. Centrate nel *centro di massa dei pixel* in un'area di 28×28

Trainig set formato da due file:

1. File con tutte le immagini (codificate come *dump* dei dati)
2. File con i valori attesi per ciascuna immagine

Soluzioni implementate

Soluzioni implementate

Soluzione ideale

[Russel03] suggerisce una rete neurale con $28 \times 28 = 748$ neuroni in input, 300 nascosti e 10 di output.

Soluzione ideale

[Russel03] suggerisce una rete neurale con $28 \times 28 = 748$ neuroni in input, 300 nascosti e 10 di output.
Uno schema per la rete è:

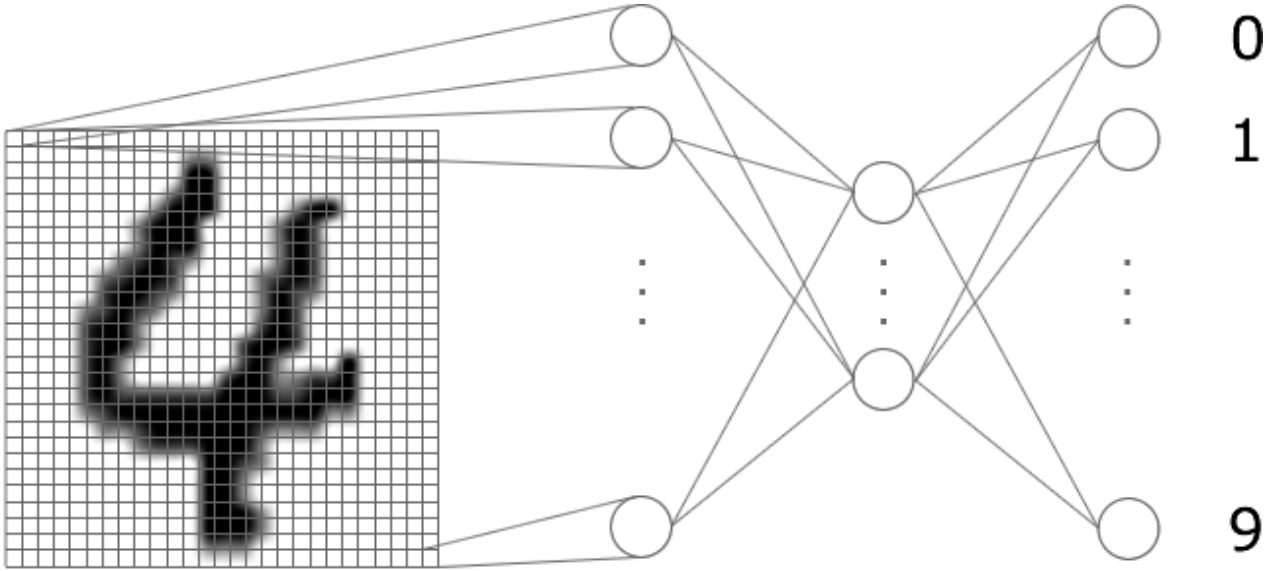


Figura: Struttura della rete implementata.

Le librerie FANN

Software *free* ed *open source* per lo sviluppo di reti neurali multilayer.

Scritte in C ma con porting in molti linguaggi, fra i quali: PHP, C++, .NET, Ada, Python, Delphi, Octave, Ruby, Prolog Pure Data e Mathematica.

Le librerie FANN

Software *free* ed *open source* per lo sviluppo di reti neurali multilayer.

Scritte in C ma con porting in molti linguaggi, fra i quali: PHP, C++, .NET, Ada, Python, Delphi, Octave, Ruby, Prolog Pure Data e Mathematica.

Algoritmi di apprendimento implementati:

1. Backpropagation incrementale
2. Backpropagation batch
3. RPROP
4. Quickprop

Le librerie FANN

Software *free* ed *open source* per lo sviluppo di reti neurali multilayer.

Scritte in C ma con porting in molti linguaggi, fra i quali: PHP, C++, .NET, Ada, Python, Delphi, Octave, Ruby, Prolog Pure Data e Mathematica.

Algoritmi di apprendimento implementati:

1. Backpropagation incrementale ← usato
2. Backpropagation batch ← usato
3. RPROP
4. Quickprop ← usato

Programmi che costituiscono il progetto

Il progetto è costituito da 5 programmi differenti:

Programmi che costituiscono il progetto

Il progetto è costituito da 5 programmi differenti:

`convert` converte le immagini e le etichette in un formato accettabili per la libreria FANN

`train` avvia la sessione di apprendimento per la rete

`test` esegue dei test con degli esempi presi dal validation set

`bulk-test` esege molti test, pescando esempi in maniera casuale dal validation set

`own-test` prende come parametro della linea di comando il nome di un file immagine, che considererà come un esempio da sottoporre alla rete

Dati per l'apprendimento

Parametri usati per l'apprendimento

Dati per l'apprendimento

Parametri usati per l'apprendimento

- Training set con 30 000 esempi

Dati per l'apprendimento

Parametri usati per l'apprendimento

- Training set con 30 000 esempi
- Massimo di 200 epoche di apprendimento

Dati per l'apprendimento

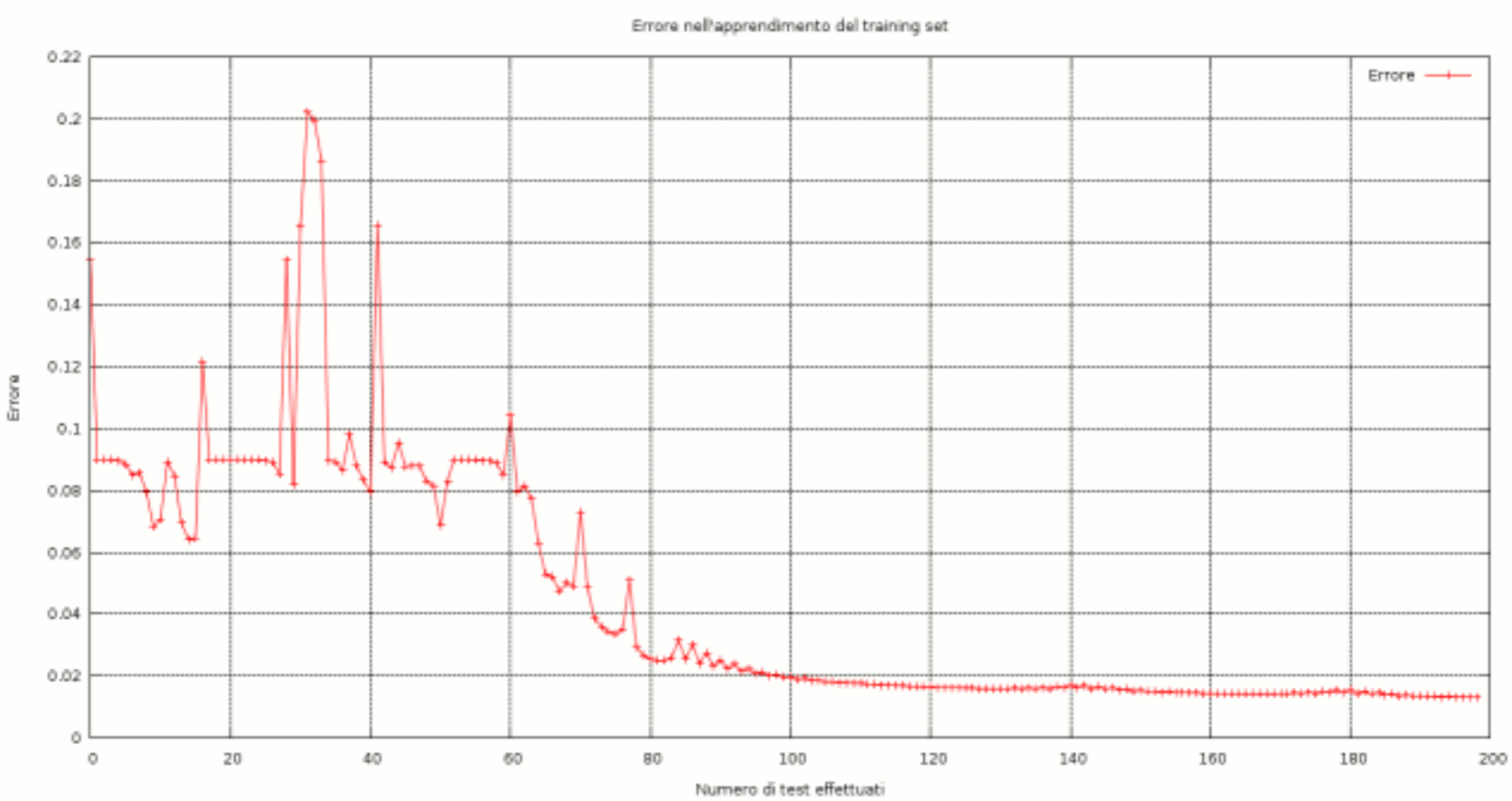
Parametri usati per l'apprendimento

- Training set con 30 000 esempi
- Massimo di 200 epoche di apprendimento
- Test su vari algoritmi:
  - Backpropagation batch
  - Backpropagation incrementale
  - Quickprop
  - Quickprop + Backpropagation batch

Risultati dell'apprendimento

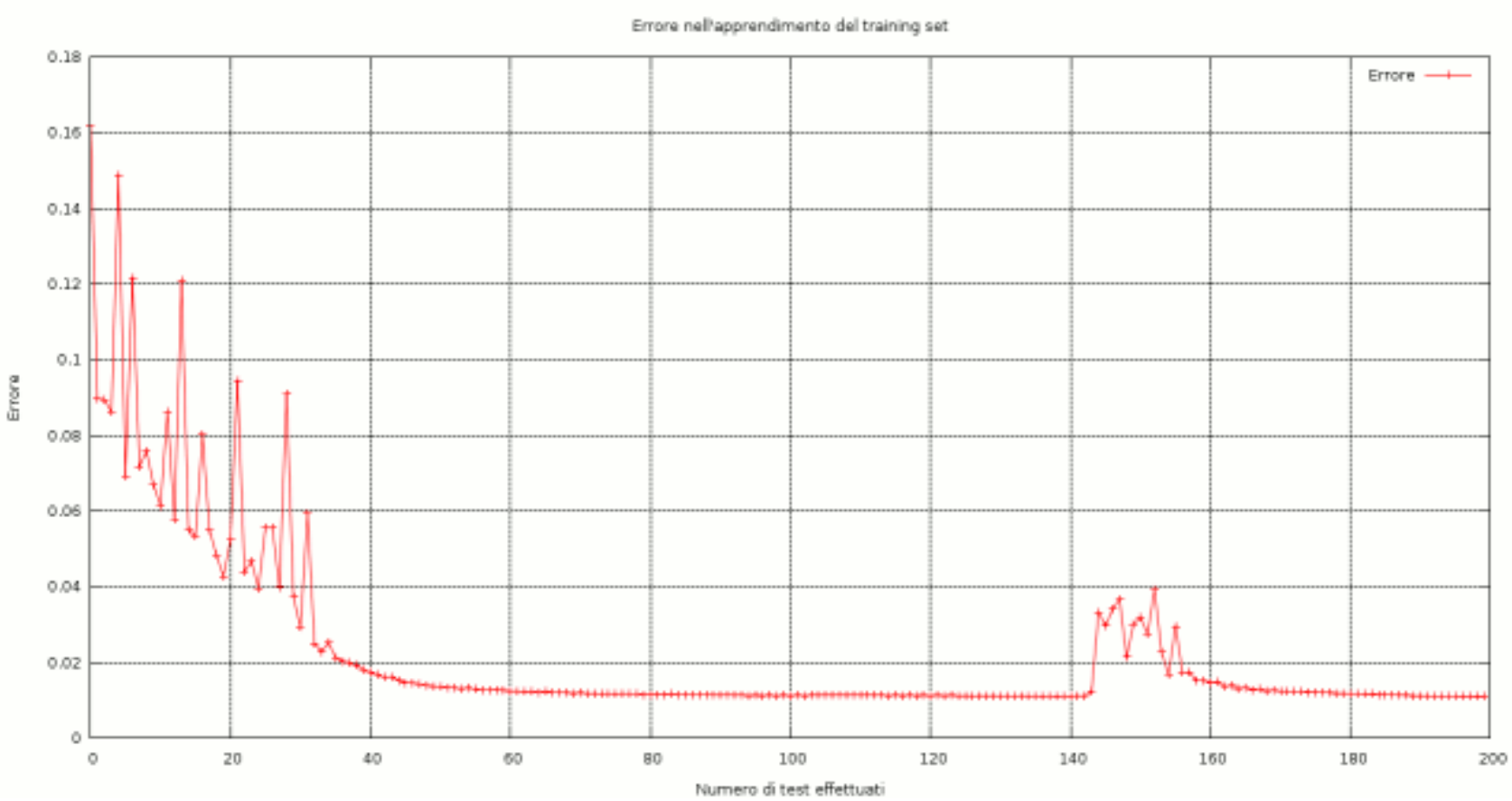
Algoritmo	**Tempo**	**MSE**	**Class.**
Backpropagation batch	1h 5m	0.01315	84%
Backpropagation incrementale	17h	0.04699	71%
Quickprop	1h 20m	0.01094	88%
Algoritmi misti	1h 20m	0.01300	92%

Andamento apprendimento con Backpropagation batch

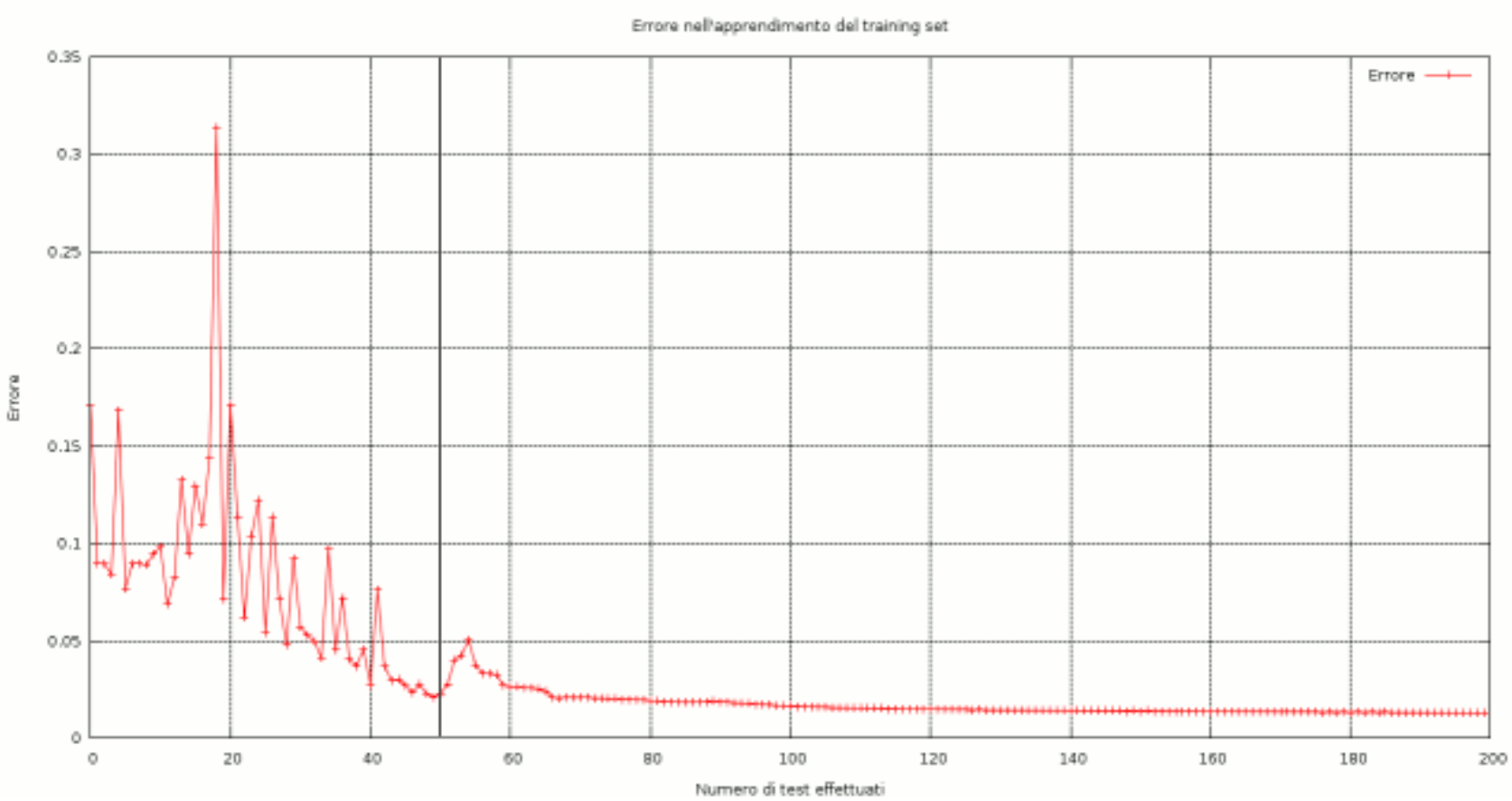


Andamento apprendimento con Backpropagation incr.

Andamento apprendimento con Quickprop



Andamento apprendimento con algoritmi misti



Esempio pratico...

I programmi in azione

Esempio pratico...

Consuntivo delle ore impiegate

Consuntivo delle ore impiegate

Consuntivo ore lavorate

In totale sono state impiegate circa 33 ore, così suddivise:

- 4 ore per studio problema
- 10 ore per sperimentazione di alcuni prodotti
- 10 ore per studio ed implementazione codice tramite FANN
- 5 ore per collaudo codice, con varie configurazioni
- 4 ore per redazione documento relazione e presentazione

Bibliografia

Bibliografia

Bibliografia

Russel03 Stuart Russell, Peter Norvig. *Artificial Intelligence: A Modern Approach (second edition)*. Prentice Hall, 2003

Fann *Fast Artificial Neural Network Library (FANN)*. `http://leenissen.dk/fann/`

Fahlman88 Scott E. Fahlman. *Faster-learning variations on back-propagation: An empirical study*. In T. J. Sejnowski G. E. Hinton and D. S. Touretzky, editors, 1988 Connectionist Models Summer School, San Mateo, CA, 1988. Morgan Kaufmann.

Mitchell97 Thomas Mitchell. *Machine learning*. McGraw-Hill Education, 1997

LeCun Yann LeCun, Corinna Cortes. *THE MNIST DATABASE of handwritten digits*. `http://yann.lecun.com/exdb/mnist/`