

DOCUMENTAZIONE DEL PROGETTO DI RETI NEURALI E VISIONE ARTIFICIALE AA 2004/2005

Possamai Lino*

Indice

1	Introduzione	1
2	Preparazione delle immagini	2
3	Rete Neurale	3
3.1	Struttura della rete	3
3.2	Campi recettivi	4
3.3	Simulated Annealing Algorithm	5
3.4	Modalità di apprendimento	6
3.5	Training	7
3.6	Criterio di arresto del training	7
4	Risultati	9
4.1	Rete Neurale	9
4.2	Simulated Annealing	10
A	HowTo del programma	10

1 Introduzione

Con questo lavoro ci siamo proposti di approfondire e di concretizzare il progetto [3] con lo scopo principale di creare un software per il riconoscimento di espressioni del volto usando una combinazione di addestramento supervisionato e non supervisionato. Il riconoscimento delle espressioni del volto è importante, per esempio, per la creazione di nuove periferiche interattive che offrono nuove possibilità agli umani di interagire con i sistemi di computer.

*matricola 800509

È stato dimostrato che il processo di riconoscimento delle facce viene effettuato in un tempo relativamente veloce: circa 100 msec. Questo risultato ci serve per affermare che l'architettura della rete neurale che più si adatta a risolvere questo tipo di problema è quella feed-forward.

La rete può essere divisa in due parti, in base al tipo di addestramento scelto. Nella prima si usa un apprendimento di tipo non supervisionato, grazie al quale riusciamo a diminuire il numero di pixel usati per rappresentare l'immagine in input. Nella seconda parte della rete viene utilizzato un addestramento di tipo supervisionato (back propagation).

Un'altra caratteristica importante della rete neurale è quella di essere formata da moduli ognuno dei quali è specializzato nel riconoscimento di una qualche espressione del volto. Nel nostro caso, si sono utilizzati tre moduli grazie ai quali si riescono a riconoscere rispettivamente l'espressione **happy**, **surprise**, **sad**. Quando nessuno di questi tre moduli è attivo, allora l'espressione riconosciuta è la **normal**.

2 Preparazione delle immagini

Le immagini utilizzate per il training del sistema sono state ricavate dal database dell'Università di Yale [2]. Queste erano composte da 15 soggetti con varie espressioni facciali e diverse illuminazioni. Come indicato nell'articolo di riferimento abbiamo selezionato solo le immagini con illuminazione normale; di queste abbiamo scelto quelle che rappresentavano le quattro pose che la rete deve riconoscere. Purtroppo uno dei soggetti aveva una foto doppia che quindi risultava inutilizzabile: i soggetti disponibili si sono perciò ridotti a 14 come riferito in [3], per un totale di 56 immagini.

Dal momento che le immagini sono di tipo PGM¹, formato molto semplice da leggere e da scrivere, si è deciso di creare un modulo del programma che fosse in grado di riconoscere questo formato.

Prima di passarle in input alla rete neurale, è stato utile che le immagini fossero preparate in modo tale che ognuna avesse le stesse caratteristiche. Con questo si intende che sono state sottoposte a diverse fasi di elaborazione quali:

- rotazione,
- ritaglio,
- ridimensionamento in una maschera di 24 per 8 pixel (vedi figura 1).

Per cercare di espandere il piccolo database Yale a nostra disposizione, abbiamo provato a fotografare alcuni soggetti con le stesse espressioni che la rete è in grado di riconoscere. Anche quest'ultime sono state elaborate secondo le fasi sopra elencate. Come chiaramente esposto in §4, il fatto di non aver potuto agire direttamente nell'impostazione delle luci nel momento di acquisizione delle immagini, ha precluso una buona capacità di riconoscimento delle stesse da parte della rete.

¹in scala di 255 livelli di grigi e con codifica ASCII

Per effettuare la manipolazione delle immagini, sono stati utilizzati due programmi, Corel Photo-Paint, incluso nella suite Corel Draw 11 e IrfanView, programma gratuito per uso personale.



Figura 1: Esempio di immagini che rappresentano l'input per la rete neurale. Ci sono 6 soggetti, ognuno dei quali è stato fotografato con 4 espressioni diverse

3 Rete Neurale

3.1 Struttura della rete

La rete neurale utilizzata per il nostro scopo, è una rete feedforward a quattro livelli, ciascuno formato rispettivamente da 192 , 48 (First Hidden Layer, **FHL**), 10 (Module Layer, **ML**) e 3 neuroni sigmoidali di cui il livello più numeroso, in termini di nodi, è rappresentato dall'input layer (**IL**) e quello meno numeroso, l'output layer (**OL**).

I primi due livelli hanno il compito di ridurre la dimensione di rappresentazione dell'immagine, preservando alcuni aspetti topologici dell'input originale. Il numero di archi di questo sottografo è uguale al numero di nodi del livello di input. La modalità con cui questi archi vengono assegnati viene stabilita in base al risultato dell'algorithm di simulated annealing applicato a tutte le immagini disponibili. Questa sottorete la chiamiamo RN_1 da qui in poi.

Il secondo ed il terzo livello² rappresentano la struttura classica delle reti feedforward in cui ad ogni nodo del livello più alto partono un numero di archi pari al numero di nodi del livello più basso (questa rete la denoteremo successivamente con RN_2).

²tutti e due si considerano *hidden layer*

Il Module Layer è costituito dai moduli di riconoscimento delle espressioni. La scelta di utilizzare una certa modularità nella rete è giustificata dal fatto che questa struttura ha portato buoni risultati quando applicata a problemi di speaker verification, face recognition, ma anche quando viene utilizzata in strumenti che consentono di verificare le ipotesi fatte sul funzionamento del cervello. Nel nostro caso, sono stati utilizzati moduli formati da tre neuroni per il riconoscimento dell'espressione surprised e happy e per il sad è stata necessaria l'aggiunta di un neurone in più dovuto al fatto che il riconoscimento di quest'ultimo risulta più difficile rispetto agli altri (vedi figura 2).

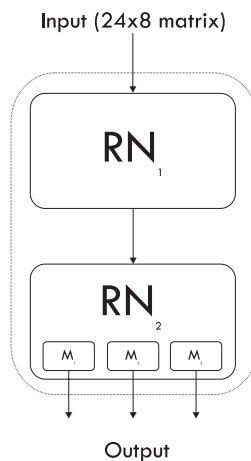


Figura 2: Schematizzazione della struttura della rete neurale. RN_1 è la rete neurale in cui l'addestramento viene fatto con la Hebb rule. RN_2 , invece, utilizza il classico algoritmo di back propagation.

3.2 Campi recettivi

In [3] gli archi della sottorete RN_1 sono stati inizializzati in modo tale che ad ogni nodo del secondo livello corrisponda un blocco di 2x2 pixel adiacenti (vedi figura 8) che appartengono all'input layer. Questi blocchi di pixel sono chiamati **campi recettivi**, definizione che richiama quella vista nelle lezioni di visione artificiale, secondo cui

un campo recettivo è quella porzione di campo visivo su cui uno stimolo luminoso produce una risposta.

I campi recettivi (che denoteremo con RF^i per indicare l' i –esimo campo recettivo) sono soggetti a cambiamento secondo la seguente dinamica:

- si seleziona un neurone dal **FHL**
- si valuta la possibilità di incrementare o decrementare il campo recettivo assorbendo o escludendo un pixel (neurone) da un campo recettivo adiacente; questo processo viene effettuato grazie all'algoritmo simulated annealing

Il simulated annealing, è un algoritmo utilizzato per la risoluzione di problemi di ottimizzazione combinatoria e per questo è necessaria una funzione costo (obiettivo) che nel nostro caso è definita nel modo seguente:

$$E(RF^i) = \left[\left(\sum_{i \in RF^i} Var_j \right) - N_i \overline{Var} \right]^2 \quad (1)$$

dove

$$\overline{Var} = \frac{\sum_{j=1}^{192} Var_j}{192}$$

e

$$Var_j = \mu[x^2] - (\mu[x])^2$$

con j che seleziona un pixel dell'immagine ritagliata, $j \in \{1, \dots, 192\}$, x è l'insieme dei valori di intensità che il pixel j assume nelle varie immagini del training set e N_i è la cardinalità del campo recettivo i .

L'energia totale del sistema sarà la somma delle energie di ogni campo recettivo,

$$\mathbf{E} = \sum_{k=1}^{48} E(RF^i) \quad (2)$$

3.3 Simulated Annealing Algorithm

Prima di presentare brevemente quali sono le caratteristiche dell'algoritmo di simulated annealing (in seguito lo indicheremo con SA), faremo un passo indietro analizzando i predecessori dell'SA: **gli algoritmi di ricerca locale**.

La caratteristica degli algoritmi che fanno parte di questa categoria è quella di effettuare un'elaborazione con lo scopo di minimizzare una funzione costo seguendo i seguenti passi:

- partono da una configurazione A ;
- eseguono una sequenza di iterazioni, ognuna delle quali consiste di una transizione dalla configurazione corrente verso un'altra che appartiene all'intorno di A ;
- se la transazione da luogo ad un miglioramento del costo, la configurazione corrente viene rimpiazzata dalla sua vicina, altrimenti una nuova configurazione tra quelle vicine viene selezionata per un nuovo confronto

Ci sono diversi svantaggi che ne derivano dall'uso di questi algoritmi, come per esempio

- il fatto che la computazione termina in un minimo locale e non si ha nessuna informazione su quanto tale minimo differisca da quello globale
- la qualità del minimo locale ottenuto dipende dalla configurazione iniziale scelta

A differenza degli algoritmi di ricerca locale, gli algoritmi di SA introducono la possibilità di accettare, con una certa probabilità p , configurazioni che forniscono un peggioramento del valore della funzione costo.

L'algoritmo di SA è in stretta analogia con il processo di solidificazione dei solidi. Infatti, il termine annealing denota un processo fisico nel quale un solido viene riscaldato progressivamente fino ad una temperatura alla quale raggiunge lo stato liquido, per poi farlo raffreddare lentamente. In questo modo, se la temperatura raggiunta è abbastanza alta e il processo di raffreddamento sufficientemente lento, tutte le molecole si dispongono in una configurazione di minima energia.

L'algoritmo 1 rappresenta la schematizzazione, in forma di pseudo codice, dell'algoritmo utilizzato. La possibilità di ottenere buoni risultati si basa sulla corretta impostazione dei parametri. Nel nostro caso, la **funzione di raffreddamento** f scelta, è la seguente: $c_{k+1} = 0.99c_k$, il **criterio di arresto** c_f scelto consiste nell'individuazione di 50 configurazioni identiche consecutive e in un valore di temperatura minimo.

Algorithm 1 Simulated Annealing

```

 $c = c_0$ 
configurazioneCorrente=INIZIALIZZA();
costo=CALCOLACOSTO(configurazioneCorrente);
repeat
  repeat
    nuovaConfigurazione=PERTURBA(configurazioneCorrente);
    /* ottiene una nuova configurazione modificando configurazioneCorrente */
    nuovoCosto=CALCOLACOSTO(nuovaConfigurazione);
    if  $\Delta(Costo < 0) \vee e^{\frac{-\Delta(Costo)}{cK_B}} > random(0, 1)$  then
      configurazioneCorrente=nuovaConfigurazione;
      costo=nuovoCosto;
    end if
  until (EQUILIBRIO());
  /* il ciclo continua fino a stabilire le condizioni di equilibrio */
   $c = f(c)$ ;
until ( $c < c_f$ )

```

3.4 Modalità di apprendimento

Una volta stabilito quale sia la configurazione migliore (intesa in termini di come sono composti i campi recettivi) che minimizza la funzione costo, il passo successivo consiste nell'assegnare agli archi, valori di inizializzazione. Si è deciso, valutando ciò che viene detto in [5], di scegliere valori che sono compresi nell'intervallo $[-0.05, 0.05]$.

Come è stato accennato in precedenza, per quanto riguarda la modalità di apprendimento della RN_1 è stato deciso di utilizzare l'Hebbian rule la quale consiste in una variazione dei pesi

secondo il criterio:

$$\Delta w_i = \eta V(\xi_i - V w_i)$$

dove w_i indica uno dei pesi che collega un neurone di input di valore ξ_i ad un neurone del **FHL** di peso netto $V = \sum_{i=1}^4 \xi_i w_i$ e con costante $\eta = 0.95$.

Nell' RN_2 i pesi degli archi, che determinano il grado di apprendimento della rete, vengono modificati utilizzando l'algoritmo di backpropagation [4] con parametro $\eta = 0.15$. Per evitare che, con la tecnica della discesa del gradiente, non ci sia convergenza in breve tempo oppure che ci sia oscillazione, si è pensato di utilizzare il termine momentum con parametro $\alpha = 0.95$.

3.5 Training

Il database a nostra disposizione è formato da poche immagini per creare un buon training set e questa caratteristica andrà ad influenzare abbastanza la capacità di generalizzazione della rete. Per questo motivo in [3] si è scelto il cross validation come modalità di testing. In questo caso si sono inclusi nel **TS**, 13 dei 14 soggetti a disposizione e il soggetto escluso è stato utilizzato come validation set, necessario per valutare la bontà della rete in presenza di facce mai viste. Questa procedura è stata ripetuta 14 volte in modo tale da scegliere quella in cui si possono ottenere migliori risultati.

Il parametro di riferimento utilizzato per osservare l'andamento della capacità di generalizzazione della rete, per decidere il momento in cui il training viene fermato e per correggere i pesi della RN_2 , è la funzione costo rappresentata da:

$$E = \frac{1}{2} \sum_{h=1}^p \|\overline{out}_h - \bar{y}_h\|_2^2 = \frac{1}{2} \sum_{h=1}^p \sum_{i=1}^n (out_i^h - y_i^h)^2 \quad (3)$$

con p cardinalità del training set e n numero di neuroni dell'output layer.

Questo termine rappresenta quanto il comportamento della rete si discosta dall'output desiderato. Il valore è stato calcolato per ogni immagine (l'operazione era necessaria anche per la modifica dei pesi degli archi) ed alla fine è stata calcolata la media rispetto alla cardinalità del TS. Questo ci ha permesso la costruzione dei grafici di figura 3 e 4 in cui si può notare l'andamento rispetto al numero di epoche dell'errore medio del TS e rispettivamente del VS.

Una scelta che è stata fatta, dedotta da una possibile interpretazione di quanto scritto in [3], è quella di evitare di effettuare la correzione dei pesi della rete quando il valore della funzione errore (3) per un'immagine è inferiore a 0.07.

3.6 Criterio di arresto del training

Per permettere un addestramento migliore e per evitare il fenomeno dell'overfitting, in presenza del quale non conviene più procedere con l'apprendimento perché si verifica un deterioramento del riconoscimento di dati mai visti, abbiamo utilizzato un criterio di arresto che consiste

- nell'utilizzare un controllo che verifica che un numero minimo di epoche siano passate prima di fermare il training,

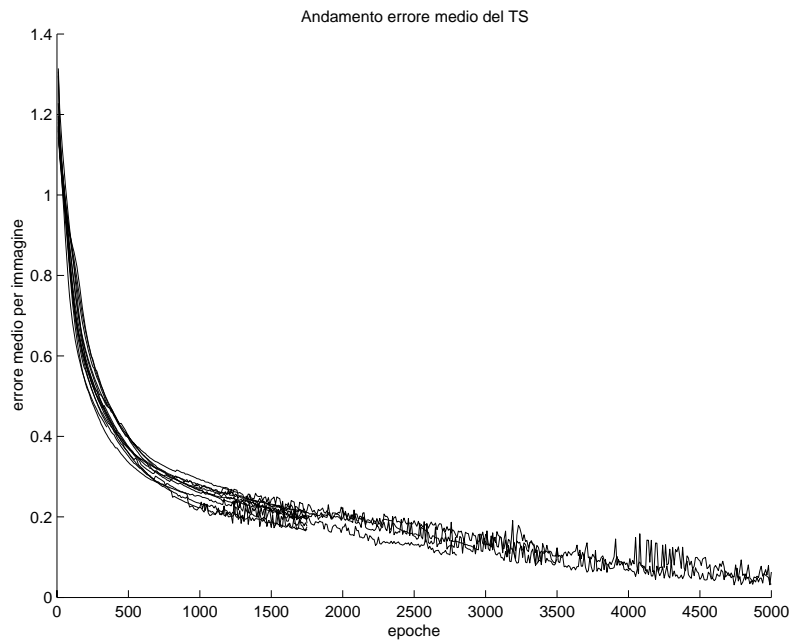


Figura 3: Andamento dell'errore medio per ogni immagine calcolato su training set differenti (cross-validation), senza alcun criterio di stop dell'addestramento

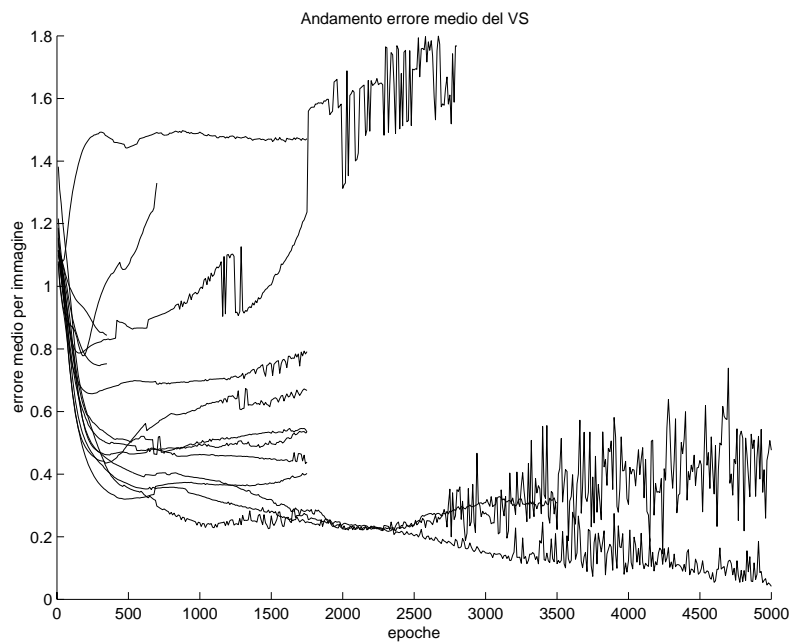


Figura 4: Andamento dell'errore medio per ogni immagine calcolato su validation set differenti (cross-validation), senza alcun criterio di stop dell'addestramento

- nel verificare che l'errore medio del TS sia inferiore all'errore medio del VS, e che il primo sia inferiore ad una soglia prestabilita,
- nell'analizzare che l'errore medio del VS assuma un andamento costante in un determinato intervallo di epoche.

Questi vincoli vengono verificati ad intervalli regolari. Se ad un punto di verifica, il valore dell'errore medio del VS è inferiore a quello memorizzato allo step precedente, allora i pesi della rete vengono salvati in una struttura in modo tale da poterli ripristinare nel momento in cui il training viene terminato.

Questi criteri sono frutto di una possibile interpretazione di quanto si riesce ad intuire in [3]. È stato comunque utilizzato un criterio più semplice, ma efficace al pari di quelli elencati sopra:

- a intervalli regolari viene calcolata la differenza tra l'errore medio del TS e l'errore medio del VS all'epoca i ($TS(i) - VS(i) = \Delta_i$) per cinque diverse epoche (consecutive) e viene fermato il training quando vale la condizione che $\Delta_i < 0, \forall i$.

4 Risultati

4.1 Rete Neurale

Il processo di ricerca dei parametri ottimali per l'algoritmo è stato molto lungo e faticoso. Sono stati provati diversi valori di inizializzazione in modo tale da avvicinarsi ad un buon comportamento della rete.

Come accennato in §3.5, solamente un soggetto (quattro immagini) è considerato come validation set e viene utilizzata la tecnica del cross-validation (CV) per effettuare il training della rete. Come si può notare dai risultati della tabella 1, i risultati variano molto in funzione del soggetto che si sceglie come VS. In definitiva, si ottengono delle percentuali di riconoscimento che spaziano da un 25% ad un massimo del 75%, anche se in alcuni casi si sono ottenuti valori sensibilmente più alti.

Questi risultati sono stati ottenuti grazie alla trasformazione da un range di validità dei pixel $[0,255]$ delle immagini di input ad uno centrato sullo zero: $[-1,1]$. Inoltre, un altro importante fattore che ha influito notevolmente è stata la scelta di usare come funzione di attivazione la tangente iperbolica, come riferito in [1].

Considerare solamente un soggetto come VS probabilmente è una scelta restrittiva. Perciò rispetto all'algoritmo principale, è stata aggiunta la possibilità di aumentare la cardinalità del VS. Anche per questo motivo sono stati aggiunti cinque nuovi soggetti che originariamente non appartenevano al database Yale, affinché la cardinalità del TS non fosse inferiore a quella del VS.

Nella tabella 2 sono presentate due tipologie di risultati che si differenziano l'una dall'altra perché nel primo caso le nuove immagini non sono inserite nel VS. L'uso del nuovo set di immagini ha provocato un degrado della capacità di riconoscimento della rete, dovuto principalmente alla diversa impostazione delle luci. Dato che non si conoscono le posizioni delle fonti di luce delle

immagini scattate alla Yale, non si possono ottenere risultati migliori se non provando a variare la percentuale di luminosità.

CV step	capacità ric.	%
14	2/4	50%
13	2/4	50%
12	3/4	75%
11	3/4	75%
10	1/4	25%
09	3/4	75%
08	3/4	75%
07	2/4	50%
06	3/4	75%
05	2/4	50%
04	3/4	75%
03	3/4	75%
02	1/4	25%
01	3/4	75%
media		61%

Tabella 1: Capacità di generalizzazione della rete neurale

4.2 Simulated Annealing

Anche in questo caso, per far funzionare in modo adeguato questo algoritmo sono state necessarie diverse prove con i parametri di inizializzazione.

La caratteristica dell'algoritmo SA è quella di cercare una configurazione di energia minima, accettando, nel corso delle iterazioni, anche delle variazioni che aumentano l'energia. Nel nostro caso, la funzione da minimizzare è la (2).

Anche se ogni esecuzione dell'algoritmo porta ad una soluzione che può essere diversa, nella figura 6 è rappresentato un classico andamento della funzione energia \mathbf{E} man mano che si procede con le iterazioni. Si nota che dopo una forte diminuzione iniziale dell'energia, segue un andamento in cui l'algoritmo cerca di sfuggire dai minimi locali per poi stabilizzarsi in un minimo globale.

A HowTo del programma

Al fine di massimizzare l'indipendenza delle varie fasi che formano il processo di riconoscimento delle immagini, si è deciso di rendere il programma più modulare possibile. Per questo motivo, (vedi figura 7) sono stati creati tre moduli, ognuno dei quali ha un compito preciso.

(a)			(b)		
$ VS $	capacità ric.	%	$ VS $	capacità ric.	%
1	3/4	75%	1	2/4	50%
2	5/8	62%	2	6/8	75%
3	7/12	58%	3	5/12	41%
4	9/16	56%	4	4/16	25%
5	10/20	50%	5	5/20	25%
6	13/24	54%	6	8/24	33%
7	13/28	46%	7	9/28	32%
8	14/32	43%	8	13/32	40%
9	19/36	52%	9	17/36	47%
10	20/40	50%	10	19/40	47%
11	21/44	47%	11	22/44	50%
12	22/48	45%	12	25/48	52%
13	24/52	46%	13	28/52	53%
14	16/56	28%	14	29/56	51%

Tabella 2: In (a) l'andamento della percentuale di riconoscimento diminuisce con l'aumentare della cardinalità del VS. In (b), invece, l'uso delle nuove immagini provocano un sensibile calo della percentuale di riconoscimento seguito da un aumento dello stesso dovuto alla minor influenza delle nuove immagini rispetto al totale

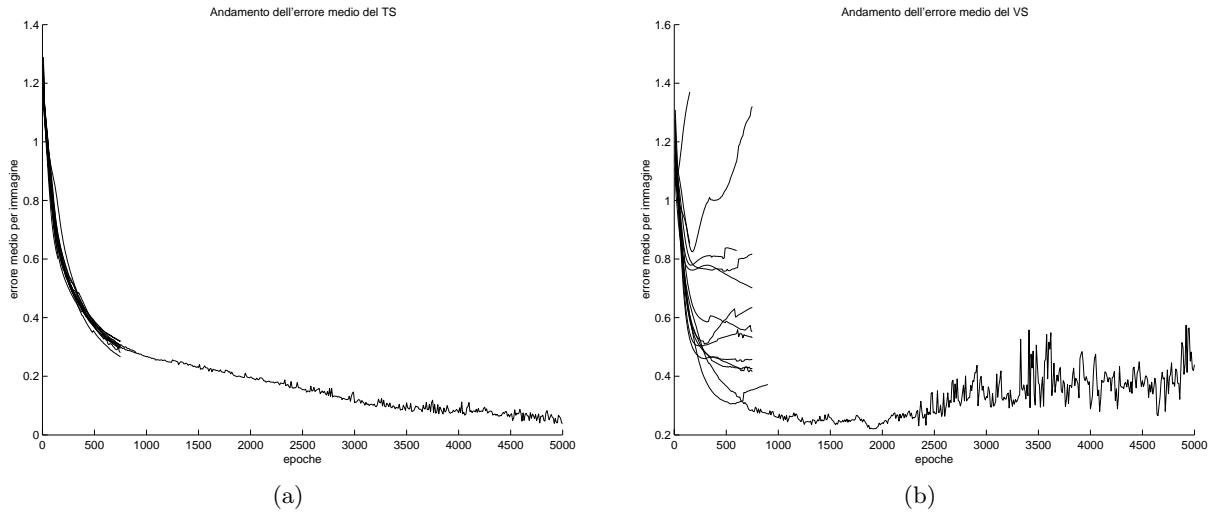


Figura 5: Classico andamento dell'errore medio per immagine del TS (a) e del VS (b), con criterio di stop Δ_i

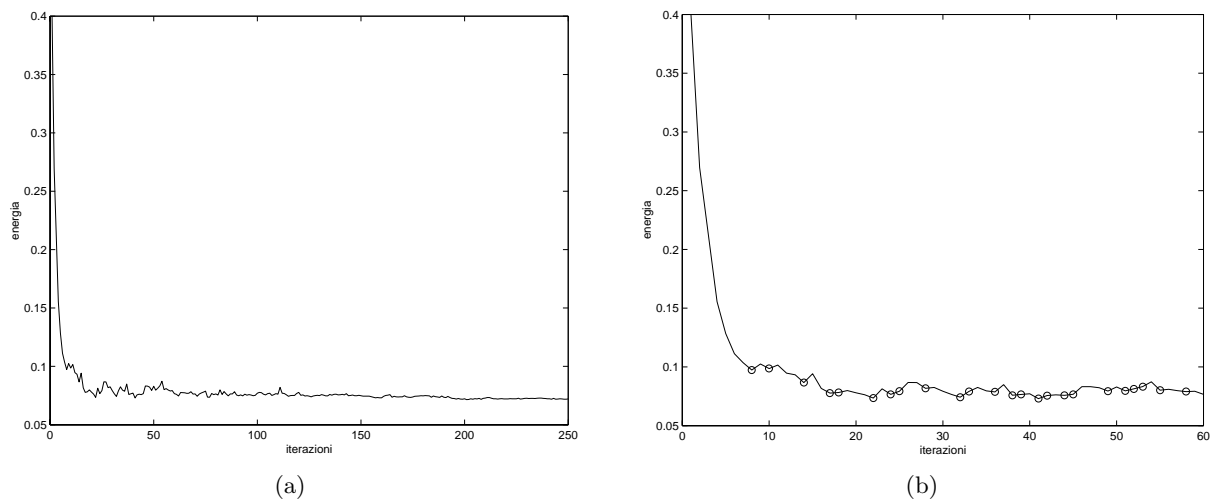


Figura 6: (a) Andamento dell'energia E al variare del numero delle iterazioni; (b) Prime 60 iterazioni del grafico di (a). Sono state altresì evidenziati i punti in cui l'algorithm accetta una configurazione che aumenta l'energia del sistema

Il primo è l'*SA*: questo modulo è incaricato di eseguire il Simulated Annealing avendo a disposizione l'insieme delle immagini e i parametri di configurazione dell'algorithm. Il risultato dell'elaborazione, rappresentato da una configurazione di costo minimo, viene salvato in un file, così da poter essere utilizzato come input nel modulo successivo. In termini di tempo di elaborazione, questo modulo può richiedere una quantità superiore rispetto a quella richiesta dagli altri. La schermata di help che viene presentata quando si esegue il programma senza nessun parametro, è la seguente:

```
Simulated Annealing Program
Utilizzo: sa.exe [opzioni]
```

Opzioni:

```
-f [file] file in cui verra' salvata la configurazione ottenuta
-l limite minimo di energia che il sistema deve avere
-h limite massimo di energia che il sistema deve avere
```

Il secondo modulo, che per semplicità chiameremo *Training*, accetterà in ingresso oltre alle immagini, la configurazione creata allo step precedente. Come suggerisce il nome, in questa fase si procede all'addestramento della rete neurale. Alla fine del training, vengono salvati su file i pesi delle RN_1 e RN_2 . La schermata di help che viene presentata quando si esegue il programma senza nessun parametro, è la seguente:

```
Training Rete Neurale
Utilizzo: trn.exe [opzioni]
```

Opzioni:

-f [file] path e nome file in cui e' salvata la configurazione
-p [dir] cartella in cui verranno salvati i file dei pesi di RN1 e RN2
-c Num scelta del validation set. Opzionale.

In questo caso, per l'ultimo parametro è necessaria una piccola spiegazione. È un parametro opzionale e se specificato, corrisponde alla selezione di uno dei 14 validation set disponibili. Se non specificato nessun valore, allora viene eseguito il cross-validation.

L'ultimo modulo, *Riconoscimento*, utilizza l'output dello step precedente (i pesi della rete), la configurazione creata dall'SA, le immagini mai viste e cerca di riconoscere le espressioni del volto fornendo in output il risultato del riconoscimento. La schermata di help che viene presentata quando si esegue il programma senza nessun parametro, è la seguente:

Riconoscimento Espressioni del Volto
Utilizzo: gen.exe [opzioni]

Opzioni:

-f [file] path e nome file in cui e' salvata la configurazione
-h [file] path e nome file in cui sono salvati i pesi della rete RN1
-b [file] path e nome file in cui sono salvati i pesi della rete RN2
-c Num scelta dell'insieme di immagini che si vuole riconoscere

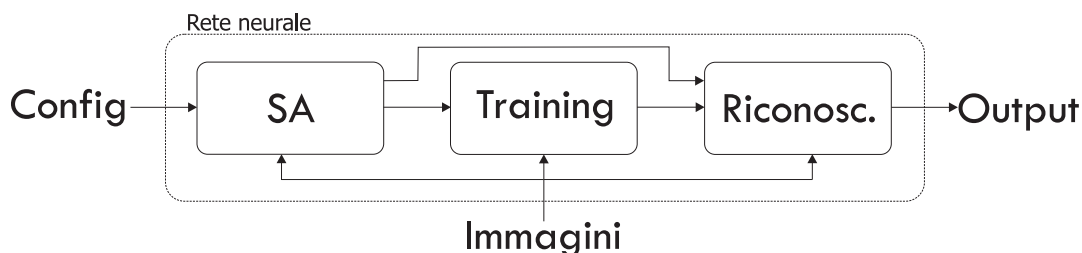


Figura 7: Modularità del programma

Riferimenti bibliografici

- [1] Ai faq, 2002. URL: <http://www.faqs.org/faqs/ai-faq/neural-nets/>.
- [2] P.N. Belhumeur and D.J. Kriegman. The yale face database, 1997. URL: <http://cvc.yale.edu/projects/yalefaces/yalefaces.html>.
- [3] Leonardo Franco and Alessandro Treves. A neural network facial expression recognition system using unsupervised local processing. *ISPA*, 2001.
- [4] Simon Haykin. *Neural Networks*. Pearson US Imports & PHIPES, 1998.

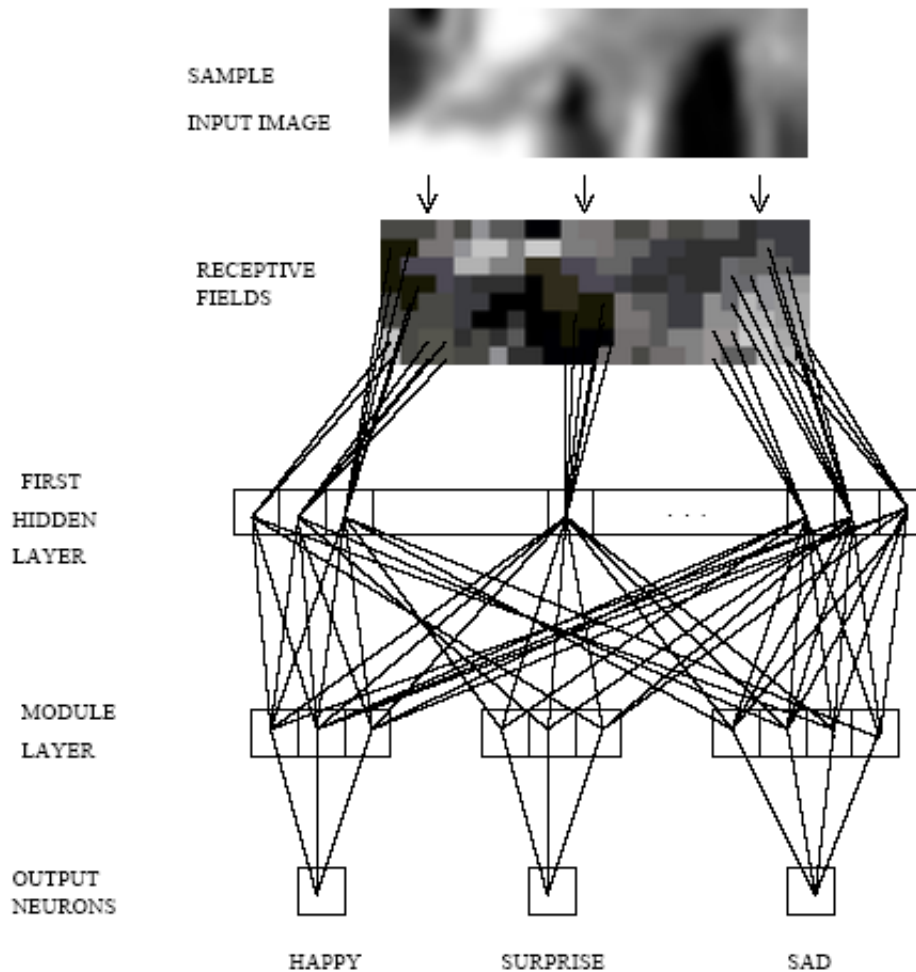


Figura 8: Struttura schematica dell'architettura della rete neurale utilizzata per il riconoscimento delle espressioni facciali. La rete ha 192 neuroni di input che corrispondono alla parte di faccia considerata i quali vengono proiettati, attraverso i pesi modificati con l'Hebbian rule, verso i 48 neuroni nel FHL, con un'organizzazione auto-adattativa dei campi recettivi. I moduli, specializzati nel riconoscimento delle diverse espressioni: happy, sad e surprised hanno una struttura con un solo livello nascosto con un output che deve essere attivato quando viene presentato in input un'immagine con un'espressione riconosciuta dal modulo.

- [5] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Perseus Publishing, 1991.

A Neural Network Facial Expression Recognition System using Unsupervised Local Processing

Leonardo Franco
Alessandro Treves
Cognitive Neuroscience Sector - SISSA
2-4 Via Beirut, Trieste, 34014 Italy
lfranco@sissa.it, ale@sissa.it

Abstract

A local unsupervised processing stage is inserted within a neural network constructed to recognize facial expressions. The stage is applied in order to reduce the dimensionality of the input data while preserving some topological structure. The receptive fields of the neurons in the first hidden layer self-organize according to a local energy function, taking into account the variance of the input pixels. There is just one synapse going out from every input pixel and these weights, connecting the first two layers, are trained with a hebbian algorithm. The structure of the network is completed with specialized modules, trained with backpropagation, that classify the data into the different expression categories. Thus, the neural net architecture includes 4 layers of neurons, that we train and test with images from the Yale Faces Database. We obtain a generalization rate of 84.5% on unseen faces, similar to the 83.2% rate obtained when using a similar system but implementing PCA processing at the initial stage.

1. Introduction

Face perception is a very important component of human cognition. Faces are rich in information about individual identity, but also about mood and mental state, being accessible windows into the mechanisms governing our emotions. Facial expression interactions are relevant in social life, teacher-student interaction, credibility in different contexts, medicine, etc. Face expression recognition is also useful for designing new interactive devices offering the possibility of new ways for humans to interact with computer systems.

From a neurophysiological point of view face recognition appears to be very important. Experiments both in monkeys and humans show the existence of dedicated ar-

reas in the brain where neurons respond selectively to faces ([11, 17, 6]). Also it has been shown that complex visual processing related to discrimination of faces is a very rapid task that can be completed in approximately 100 msec suggesting the involvement of a feed-forward neural mechanism [13].

In this work we construct a modular neural network including two parts, trained in different ways: first, a hidden layer of neurons having the task of reducing the dimensionality of the data to a more suitable form, to be classified by specialized modules, sometimes called "experts", of the second part, trained with backpropagation.

Unsupervised algorithms have been shown to be very effective in the task of reducing the high dimensionality of the input data, for example PCA or ICA algorithms [15, 2]. In our work an unsupervised procedure is applied locally, preserving some topology of the original images. Our choice is motivated by biological considerations based on the idea that a network will operate better if the variance in the input representation is distributed across many input neurons, and not just to a few, as a PCA algorithm tends to do [19].

On the other hand, modularity appears to be a very effective solution to complicated tasks allowing better generalization properties, reducing the longer training times, and being also adaptive [7, 8]. Modular networks have been used successfully in several tasks such as speaker verification, face recognition, time series prediction, etc. [3, 4, 18], being also very useful tools for exploring hypothesis about brain function [5, 10].

Different systems have been constructed to deal with facial expressions, see for instance [14] and references therein, but few of them use a neural network approach. For example, in [16] a feedforward network with PCA input encoding of some facial features (eyes and mouth) is trained to classify emotions, obtaining an 84 % generalization rate; Lisetti & Rumelhart [14] have constructed a backpropagation network to classify the degree of expressive-

ness of faces. Our work continues to explore the potential of neural networks to perform this kind of task, trying to respect some biological constraints, using the capabilities of modular systems, and reducing to a minimum the preprocessing stage.

2. The Database of Images

The Yale Face Database [1] contains 165 gray scale images in GIF format of 15 male individuals of different races and features (glasses, beard, moustache, etc.). There are 11 images per subject including different expressions, views, illumination condition, etc.

We use a subset of the database that consists of 14 subjects displaying 4 facial expressions: neutral face, happy, sad and surprised. The images were cropped to obtain input images 8 pixels width by 24 pixels height covering a portion of the face located on the left side. (See Figure 1). The images were centered taking the tip of the nose as reference and some light illumination correction was applied to a couple of images; both operations were carried out by a human observer. The resulting images were transformed to pgm 8-bit gray scale format, ready to be fed into the network after a linearly scaled transformation of pixel intensity to the interval $[0, 1]$.

Figure 1 shows a sample of the different expressions displayed by one of the subjects. In the leftmost image the area of the face cropped and used as input is shown.



Figure 1. Sample subject showing the four full face expressions (neutral, happy, sad and surprised). The white rectangle inside the rightmost figure corresponds to the area cropped and used as input for the neural network.

3. Network Structure

The network consists of a 4 layer modular neural structure composed of sigmoidal units. The input layer has 192 units corresponding to the 24×8 pixels of the area cropped from the original images. Every input neuron transmits information through a single hebbian weight, projecting to a specific neuron in the first hidden layer, selected according to a self-organized process. Thus one has at this level a new

reduced representation of the images, 48 neurons, that preserves some topological aspects of the original input. The whole network architecture is shown in figure 2, where at the top we show a sample input image followed by the structure of the receptive fields corresponding to the first hidden layer neurons.

After this unsupervised compression the network splits into three modules corresponding to the expressions different from the neutral face: happy, sad and surprised. The structure of the modules could depend on the emotion they specialize in; in the case we consider here they have all the same type of architecture: one hidden layer fully connected with one output unit. There is a difference in the number of hidden neurons belonging to each modules since the recognition of happy and surprised faces is much easier than the recognition of sad ones, a fact that was previously known from experiments both with humans and computers [5]. It was necessary to put 4 hidden neurons for sad faces while 3 neurons were enough for happy and surprised ones. In this way the output of the whole network has 3 neurons that should be all OFF when a neutral face is presented, while when a face displaying an emotion is shown, the corresponding module unit should be ON.

Table 1. Generalization error rates for the modules, specialized in happy, sad and surprise faces, and for the whole net using first layer self-organizing receptive fields-hebbian (SORF-Hebbian), PCA, and random processing. The generalization error is measured after the training procedure succeeds, when the training error per example turns out to be around 0.02.

Expression Module	Error (SORF-Hebbian)	Error (PCA)	Error (Random)
Happy	0.057	0.082	0.089
Sad	0.044	0.032	0.154
Surprise	0.053	0.053	0.071
Total	0.154	0.167	0.314

3.1. Self-organization of receptive fields

As we mentioned before, the first layer of weights self-organizes according to the variance of the input pixels, with the aim of obtaining a distributed activity in the first hidden layer of neurons. Initially, the receptive fields of first hidden layer neurons are square blocks of 2×2 pixels of the input images.

The receptive fields evolve according to the following dynamics: a neuron from the first hidden layer is selected,

then the convenience of increasing its receptive field size, by absorbing a weight from a contiguous pixel at the input level, is computed. A possible modification of the receptive field is evaluated through a simulated annealing procedure [12] involving a local energy function calculated for the two receptive fields involved, one that may increase, that we call RF^+ and the other that could decrease its size, RF^- . The change in the configuration is always accepted if the energy difference between initial and modified states is negative but could also be accepted in case the difference is positive, depending on a time decreasing probability. We define the energy of the receptive fields, $E(RF^i)$, as:

$$E(RF^i) = \left[\left(\sum_{j \in RF^i} Var_j \right) - N_i \overline{Var} \right]^2,$$

where Var_j is the variance of a pixel belonging to the receptive field i under consideration, \overline{Var} is a constant that we set to the mean variance of all the pixels, and N_i is the number of pixels forming the receptive field.

Through this procedure we obtain new reorganized receptive fields with a variable dimension that we constrain between 1 to 10 pixels. The dynamics tends to form small receptive fields containing pixels with high variance and larger receptive fields including many low variance pixels. In this way more importance is given to pixels with a higher variance but at the same time it is possible to obtain a distributed response, at the first hidden layer units, by clustering many low variance pixels in some receptive fields.

In figure 3 are shown three states corresponding to initial, intermediate and final stages of the self-organizing process. Different colour mean a different receptive field, the mean variance of the constituting pixels being indicated by the brightness: the darker tones correspond to a lower mean variance. Note that the size of the darker clusters is larger compared to the clearer ones.

4. Training procedure

As the amount of data available for training and testing is limited (14 subjects, 56 images), we decided to use a cross-validation test, normally used in similar situations. In this procedure 13 out of the 14 available subjects are chosen to train the network and the 4 unseen faces of the remaining subject are used to test the generalization ability of the system. This procedure is repeated 14 times, one for each subject being kept out of the training set.

The first layer of 192 weights, one for each input pixel, is trained with an unsupervised hebbian algorithm.

The Hebbian rule used is Oja's rule, known to tend to a principal component analysis of the input vectors, converging to the largest eigenvector, while normalization is

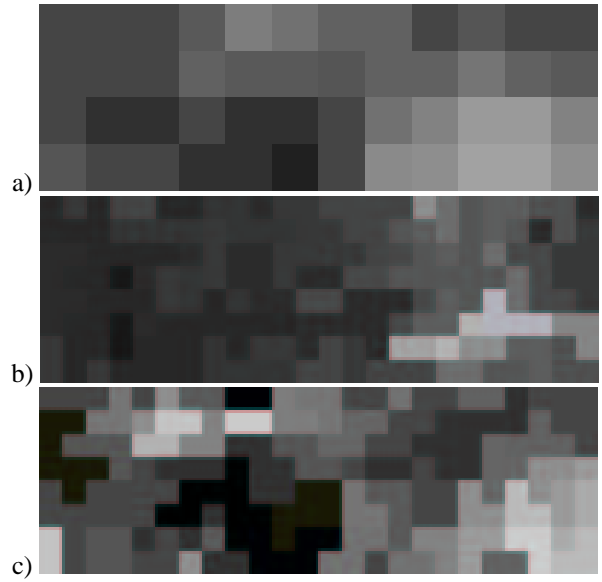


Figure 3. Receptive field evolution at different stages of the self-organizing process a) Initial state b) Intermediate state c) Final state. The mean variance of the input pixels corresponding to the receptive fields is indicated by the level of brightness, with darkest pixels being the low variance ones.

ensured [15, 9]. The change in the weight values can be written as,

$$\Delta w_i = \eta V (\xi_i - V w_i), \quad (1)$$

where w_i is one of the weights connecting an input neuron with value ξ_i to a first hidden layer neuron, with net input V ,

$$V = \sum_{i=1}^4 \xi_i w_i, \quad (2)$$

and η is a learning constant that was kept fixed to 0.05.

The rest of the weights, those belonging to the modules, were trained with the standard backpropagation algorithm (Hertz, Krogh & Palmer, 1993; Haykin, 1995). To prevent overtraining and to permit a better generalization capacity we monitor the training error on each input image, to stop the training on such image when this error is lower than 0.10. Since the backpropagation training is an on-line procedure, at the end of the training phase the average error per example is decreased to 0.02, approximately.

All layers of weights were trained at the same time upon the presentation of an input image.

5. Results and Discussion

We explore the generalization ability of a modular neural system to classify facial expressions. Using a mixed learning scheme composed by unsupervised-supervised training, we obtain a generalization ability on novel faces of 84.5%, compared to a 83.2% when replacing the unsupervised procedure by a principal component (PCA) one. The generalization error rates produced by the three modules specialized in different expressions are shown in table 1. We compared the results from our mixed "Self Organized Receptive Fields - Hebbian" (SORF-Hebbian) scheme to those obtained replacing the unsupervised process with PCA preprocessing, and also with a set of random weights and receptive fields in the initial configuration (see 3a). For the case of using the PCA analysis we project the input data onto the N principal component, and train the backpropagation modules with the resulting data. We experiment with different values of N , obtaining the best results with $N = 36$. The random processing case corresponds to setting the weights of the first layer to random values uniformly within the range $[0 - 0.7]$, while no learning is applied. This procedure has shown to perform better than the simple average of weights, and it is shown here just for comparison.

Self organization and Hebbian learning, in our case applied locally, confirm to be interesting procedures for compressing data in a more biological way, compared to a PCA approach.

The advantage of the modular approach is that it permits the addition of new modules to recognize different expressions that could be trained separately.

We are currently considering many possible extensions of this work, trying to implement the unsupervised processing stage through a competitive learning scheme and also to test the system with a more extensive database. It would also be desirable that the network itself should be capable of performing the identification of a face in a complex input image, permitting the use of the system in a more realistic way, for example to mount it on a robot.

6. Acknowledgments

We acknowledge partial support from the Human Frontiers Program Grant RG0110/1998-B.

References

- [1] P. N. Belhumeur and D. J. Kriegman. The yale face database. URL: <http://cvc.yale.edu/projects/yalefaces/yalefaces.html>, 1997.
- [2] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1997.
- [3] Y. Bennani. Multi-expert and hybrid connectionist approach for pattern recognition: speaker identification task. *Intl. Journal of Neural Systems*, 5(3):207–216, 1994.
- [4] M. N. Dailey and G. W. Cottrell. Organization of face and object recognition in modular neural network models. *Neural Networks*, 12(7-8):053–1074, 1999.
- [5] M. N. Dailey, G. W. Cottrell, and R. Adolphs. A six-unit network is all you need to discover happiness. In *Twenty-Second Annual Conference of the Cognitive Science Society*, 2000.
- [6] R. Desimone. Face selective cells in the temporal cortex of monkey. *Journal of Cognitive Neuroscience*, 3:1–8, 1991.
- [7] L. Franco and S. A. Cannas. Generalization properties of modular networks implementing the parity function. *IEEE Transactions in Neural Networks (In Press)*, 2001.
- [8] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan/IEEE Press, 1994.
- [9] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 1991.
- [10] R. A. Jacobs. Nature, nurture, and the development of functional specializations: a computational approach. *Psychonomic Bulletin & Review*, 4(3):299–309, 1997.
- [11] N. Kanwisher, J. McDermott, and M. M. Chun. The fusiform face area: A module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience*, 17:4302–4311, 1997.
- [12] S. Kirpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, pages 671–680, 1983.
- [13] S. R. Lehky. Fine discrimination of faces can be performed rapidly. *Journal of Cognitive Neuroscience*, 12(5):848–855, 2000.
- [14] C. L. Lisetti and D. E. Rumelhart. Facial expression recognition using a neural network. In *Proceedings of the 11th International Florida Artificial Intelligence Research Society Conference (FLAIRS'98)*, Menlo Park, CA, 1998.
- [15] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(1):61–68, 1989.
- [16] C. Padgett and G. W. Cottrell. A simple neural network models categorical perception of facial expressions. In *Proceedings of the Twentieth Annual Cognitive Science Conference*, Madison, WI, Mahwah: Lawrence Erlbaum, 1998.
- [17] D. I. Perret, E. T. Rolls, and W. Caan. Visual neurons responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, 47:329–342, 1982.
- [18] V. Petridis and A. Kehagias. *Predictive Modular Neural Networks: Applications to Time Series*. Kluwer Academic Publishers, Boston, 1998.
- [19] E. T. Rolls and A. Treves. *Neural Networks and Brain Function*. Oxford, 1998.