Progetto Intelligent Systems

Università di Pisa

Antonio Le Caldare, Vincenzo Consales

year 2017/2018

Contents

Chapter 1

Parte I

1.1 Generazione delle copie

In questa prima fase abbiamo generato delle copie degli spettri master in modo tale da poter allenare la rete a riconoscere le differenze tra gli spettri master e copia nella fase succesiva. Affinchè siano rispettati i requisiti, è necessario che gli spettri copia siano solo lievemente differenti dagli originali, di modo che il colore rapresentato dalla copia sia molto simile. Per fare ciò abbiamo preso in considerazione varie tecniche, tra le quali:

- sommare un numero random a tutti i campioni dello spettro
- moltiplicare tutti i campioni per un numero random
- dividere lo spettro in 3 fasce, generare 3 numeri random, sommare il numero random a tutti i campioni della fascia corrispondente
- dividere lo spettro in 3 fasce, generare 3 numeri random, moltiplicare il numero random per tutti i campioni della fascia corrispondente

Tutti i metodi sono risultati efficaci a patto di regolare oppurtunamente il range di generazione dei coefficienti randomici. L'addizione e la moltiplicazione sono sostanzialmente equivalenti, anche se i coefficienti hanno un diverso peso sul rumore. Per generare le copie abbiamo deciso di utilizzare il rumore moltiplicativo. Abbiamo deciso di utilizzare tutte le alternative allo scopo di avere copie generate in modo variegato.

Per ogni colore del set master generiamo 10 copie, di cui 5 sono disturbate moltiplicando tutti i campioni dello spettro per un numero random piccolo, mentre le restanti 5 sono disturbate su 3 fasce diverse dello spettro. Infine sommiamo un rumore a media nulla molto piccolo a tutti i campioni degli spettri copia. Il codice seguente lo copie dello spettro disturbato *specnoisedP* a partire da *specmasterP*.

```
for i_spec = 1:1269
    for sampl = 1:copies
        specmasterP(:, (i_spec-1)*copies + sampl) = spectra(:, i_spec);

        if( sampl <= floor(copies/2) )
            a = random('unif',0.95,1.12);
            specnoisedP(:, (i_spec-1)*copies + sampl) = spectra(:,
                i_spec) * a;
        else
            %DIVIDE BY 3
            a = random('unif',0.97,1.12);
            b = random('unif',0.97,1.12);
            c = random('unif',0.97,1.12);
            specnoisedP(1:146, (i_spec-1)*copies + sampl) = spectra
                (1:146, i_spec) * a;
            specnoisedP(147:300, (i_spec-1)*copies + sampl) = spectra
                (147:300, i_spec) * b;
            specnoisedP(301:421, (i_spec-1)*copies + sampl) = spectra
                (301:421, i_spec) * c;
        end
        r = -0.0005 + 0.001 * rand(421,1);
        specnoisedP(:, (i_spec-1)*copies + sampl) = specnoisedP(:, (
            i_spec-1)*copies + sampl) + r;
    end
end
```

Forniamo alcuni esempi significativi di coppie di colore generate (originale, copia).

Dopo una verifica visiva possiamo monitorare di quanto le copie siano state disturbate, in modo aggregato, rispetto al colore master. Per calcolare la differenza tra la copia e master è necessario convertire gli spettri in coordinate L*a*b ed infine calcolare le distanze euclidee.

$$\Delta E^* = \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2} \tag{1.1}$$

Per convertire gli spettri in coordinate L*a*b abbiamo usato la funzione *roo2lab*() del tool optrop per MATLab ed infine abbiamo calcolato le distanze euclidee tramite la funzione *de*() . Possiamo rappresentare la distribuzione delle distanze in un istogramma.

Dal grafico è possibile notare come le coppie stiano nel range 0-4 circa, confermando il fatto che le copie generate siano più o meno simili a quelle originali, come richiesto dalle specifiche.

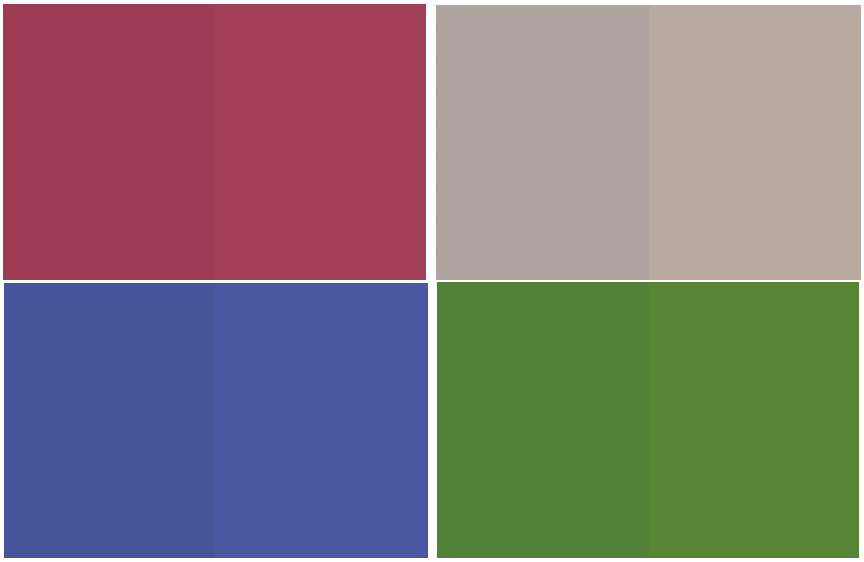

Figure 1.1: Esempi di copie generate (originale, copia)

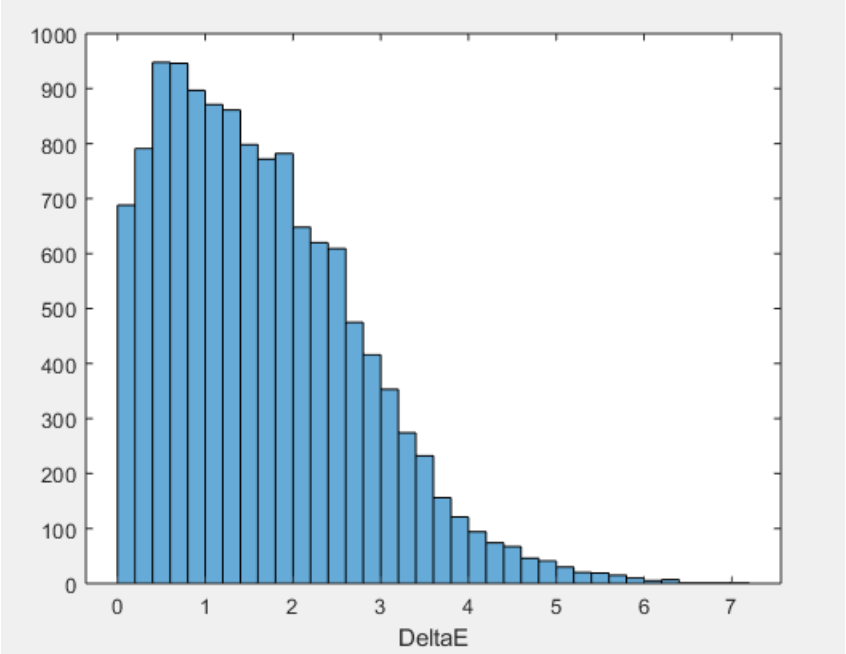

Figure 1.2: Istogramma delle DeltaE delle coppie (master, copia disturbata)

1.2 Feature Selection/Extraction

Affichè la rete riconosca lo spettro ma sia anche efficiente è necessario che quest'ultimo sia compresso e riassunto da delle features. Ricordiamo infatti, che lo spettro è rappresentato tramite 421 campioni, pertanto vanno selezionate le caratteristiche essenziali in modo da abbattere gli ingressi da fornire alla rete.

A questo punto dobbiamo scegliere quale feature estrarre. Abbiamo ragionato principalmente sul materiale fornito dalle specifiche, in particolare su come viene calcolato il colore percepito dall'occhio umano secondo gli standard CIE. Sono state prese in considerazione diverse funzioni, cioè *mean, mode, median, var, skewness, min and max (sia massimi/minimi che punti di massimo/minimo).* La *feature selection* è stata effettuata dividendo lo spettro in 6 parti uguali e applicando, per ognuna di queste, le funzioni prima elencate. Pertanto ogni funzione fornisce 6 features rappresentative dello spettro considerato.

Il metodo utilizzato per la *feature extraction* è quello fornito da MATLab attraverso la funzione *sequentialfs().* Il numero di feature estratte è stato oggetto di analisi al fine di individuare il minor numero di feature, in modo tale da rendere la rete efficiente, e ottenere le migliori prestazioni in termine di errore quadratico medio. Dopo attente analisi abbiamo deciso di selezionare 12 features visto che queste risultano sufficienti per ottenere un errore quadratico medio pari a 0.01.

1.3 Risultati

La rete che ha i risultati migliori è quella con 12 neuroni nello strato di ingresso e 8 nell'hidden layer (unico strato). Abbiamo eseguito la prova per 10 ripetizioni ed abbiamo ottenuto un *Mean Squared Error* pari a circa 0.017 e un valore *R (Regression)* pari a 0.99338 ± 0.0004.

Feature estratte Le feature estratte dalla *sequentialfs* sono state le seguenti:

master: mean(2), mode(3), median(4), median(5), skewness(3), maxYValue(2), minYValue(1)
copie: mean(2), var(2), mode(1), skewness(3), maxYValue(2)

(1,2,3) indicano i 3 range dello spettro equidimensionati sulla quale le funzioni sono applicate
(4,5) sono 2 range aggiuntivi, che si sovrappongono ai precedenti (dividono lo spettro in 2)

Tuttavia differenti esecuzioni hanno portato, spesso, all'estrazione di feature differenti: questo può essere dovuto alla correlazione tra le varie funzioni scelte (es. la

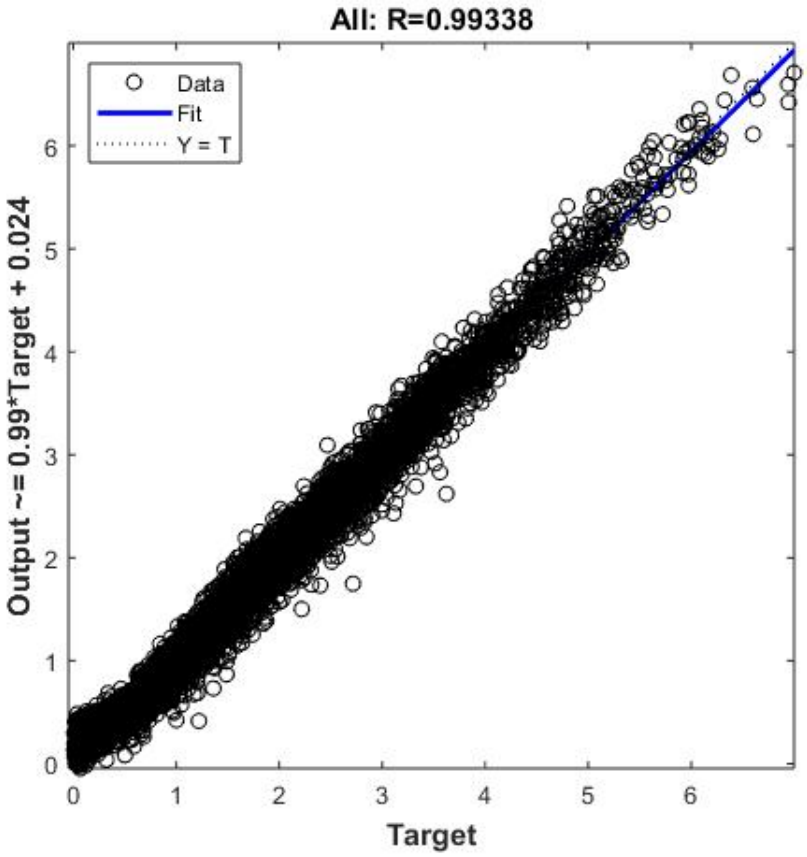

Figure 1.3: Retta di Regressione della rete neurale

skewness è calcolata dal mean e dalla varianza).

Chapter 2

Parte II

Nella seconda parte delle progetto abbiamo sviluppato una rete neurale che non si limita ad approssimare la formula 1.1 ma tiene conto anche delle diffenze percettive dell'occhio umano. L'equazione infatti è imprecisa per certe zone dello spazio L*a*b e pertanto l'output della rete neurale precedente va corretto. Per fare ciò è necessario passare dallo spazio di colore CIE L*a*b allo spazio di colore CIE L*C*h ed infine definire delle regole di inferenza per un sistema fuzzy di tipo Madamani. Ricordiamo gli intervalli delle coordinate nello spazio L*C*h.

- *Lightness (L^*)* Range [0, 100]
- *Hue (h)* Range [0, 360°]
- *Chroma (C^*)* Il range dipende dal valore L^*. $C_{\max} = 127$ quando $L^* = 50$ cioè quando si è al centro dell'ellissoide descritto dall spazio L*C*h mentre è $C^* = 0$ quando $L^* = 0$ oppure $L^* = 100$. La relazione tra L^* e C^* quindi si può descrivere come un ellisse.

$$\frac{C^2}{127^2} + \frac{(L-50)^2}{50^2} = 1 \tag{2.1}$$

Quindi $C'_{\max}$ si può calcolare come

$$C'_{\max} = 127\sqrt{1 - \frac{(L-50)^2}{50^2}} \tag{2.2}$$

Pertanto definiamo

$$c\% = 100\frac{C}{C'_{\max}} \tag{2.3}$$

Il range di c% è [0, 100] ed è questo ultimo valore che useremo per le regole di inferenza.

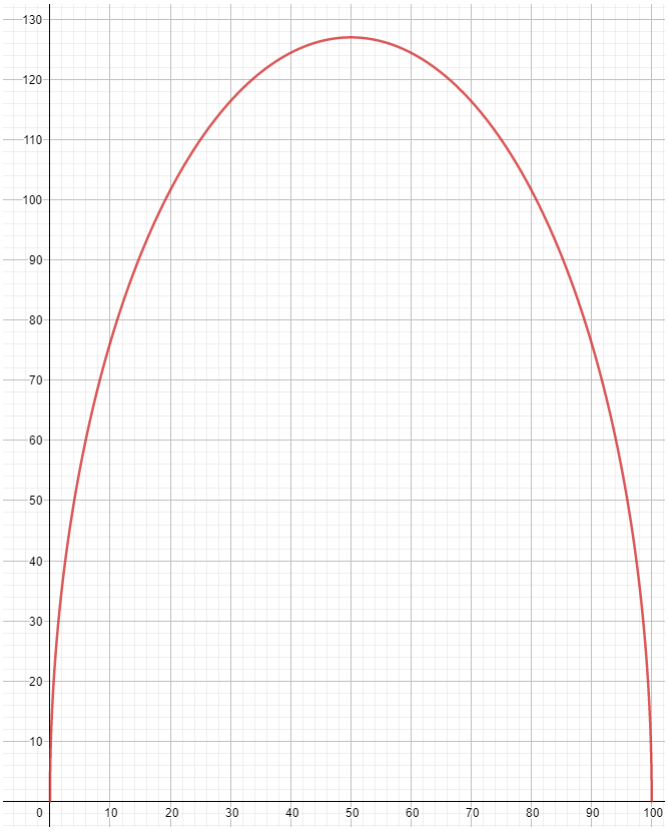

Figure 2.1: Grafico del C'_{max} al variare di L

2.1 Membership function

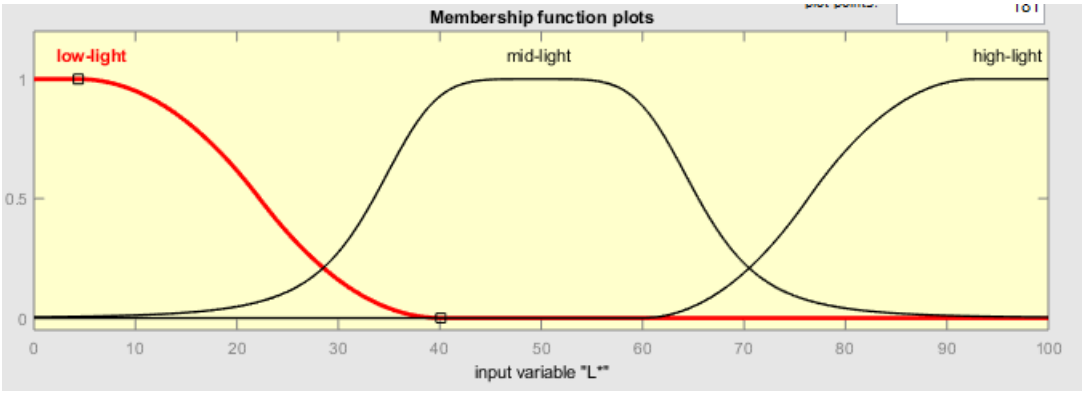

Figure 2.2: Funzione di membership Light

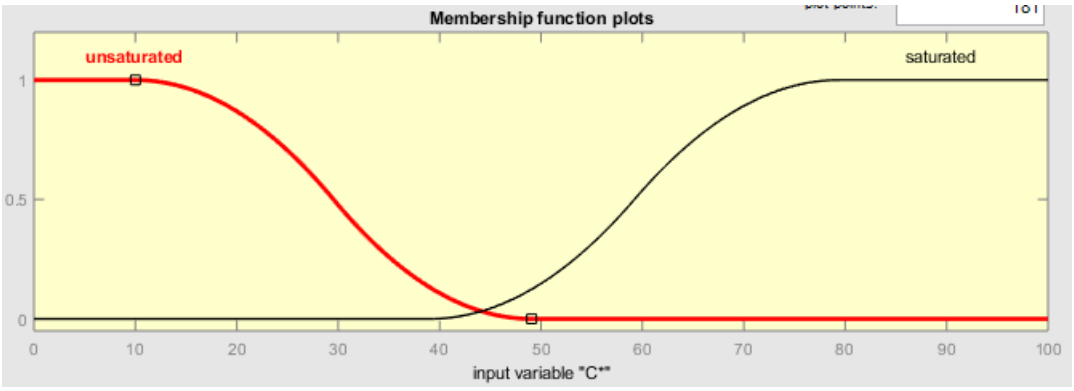

Figure 2.3: Funzione di membership Saturation

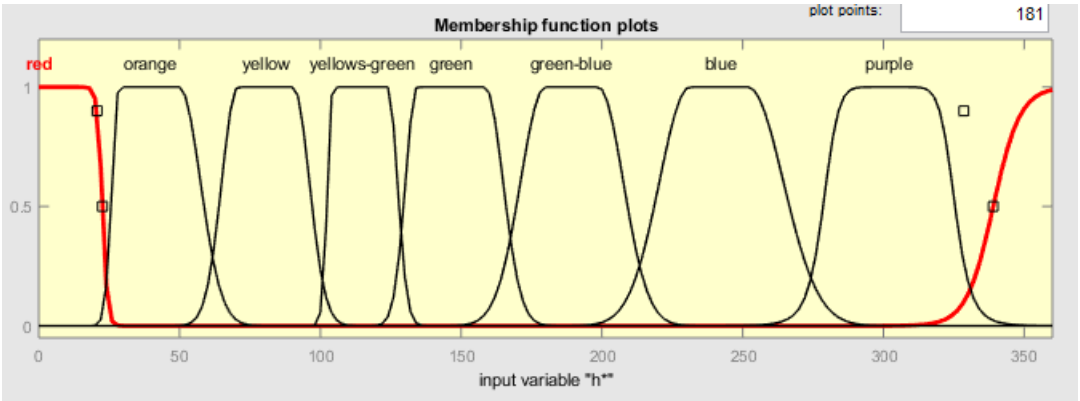

Figure 2.4: Funzione di membership Hue (Colore)

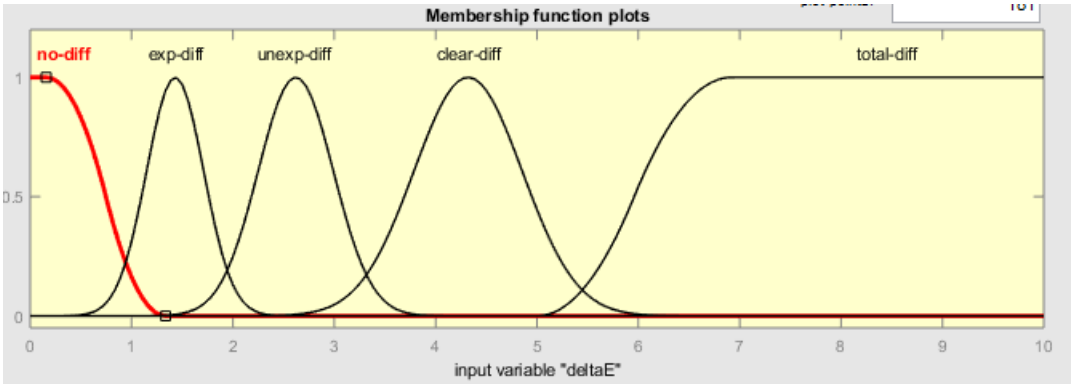

Figure 2.5: Funzione di membership DeltaE (distanza euclidea)

2.2 Regole

1. If (L* is mid-light) and (C* is saturated) and (h* is orange) and (deltaE is unexp-diff) then (correctedDeltaE is exp-diff) (1)
2. If (L* is high-light) and (C* is saturated) and (h* is yellow) and (deltaE is unexp-diff) then (correctedDeltaE is no-diff) (1)
3. If (L* is high-light) and (C* is saturated) and (h* is yellow) and (deltaE is clear-diff) then (correctedDeltaE is exp-diff) (1)
4. If (L* is mid-light) and (C* is saturated) and (h* is green) and (deltaE is unexp-diff) then (correctedDeltaE is exp-diff) (1)
5. If (L* is mid-light) and (C* is saturated) and (h* is green) and (deltaE is exp-diff) then (correctedDeltaE is no-diff) (1)
6. If (L* is mid-light) and (C* is saturated) and (h* is blue) and (deltaE is clear-diff) then (correctedDeltaE is exp-diff) (1)
7. If (deltaE is no-diff) then (correctedDeltaE is no-diff) (0.5)
8. If (deltaE is total-diff) then (correctedDeltaE is total-diff) (0.2)
9. If (deltaE is clear-diff) then (correctedDeltaE is clear-diff) (0.2)
10. If (deltaE is unexp-diff) then (correctedDeltaE is unexp-diff) (0.2)
11. If (deltaE is exp-diff) then (correctedDeltaE is exp-diff) (0.3)
12. If (L* is low-light) and (deltaE is clear-diff) then (correctedDeltaE is exp-diff) (1)
13. If (L* is low-light) and (deltaE is unexp-diff) then (correctedDeltaE is no-diff) (1)
14. If (L* is low-light) and (deltaE is exp-diff) then (correctedDeltaE is no-diff) (1)
15. If (L* is low-light) and (deltaE is total-diff) then (correctedDeltaE is exp-diff) (1)
16. If (L* is not low-light) and (C* is unsaturated) and (deltaE is not no-diff) then (correctedDeltaE is exp-diff) (1)
17. If (L* is not low-light) and (C* is unsaturated) and (deltaE is no-diff) then (correctedDeltaE is no-diff) (1)

Figure 2.6: Funzione di membership DeltaE (distanza euclidea)