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.

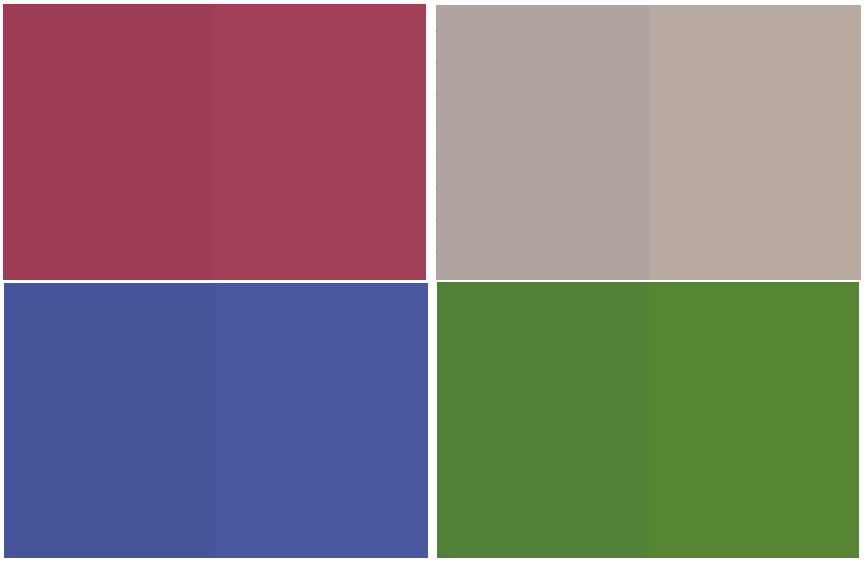

Figure 1.1: Esempi di copie generate (originale, copia)

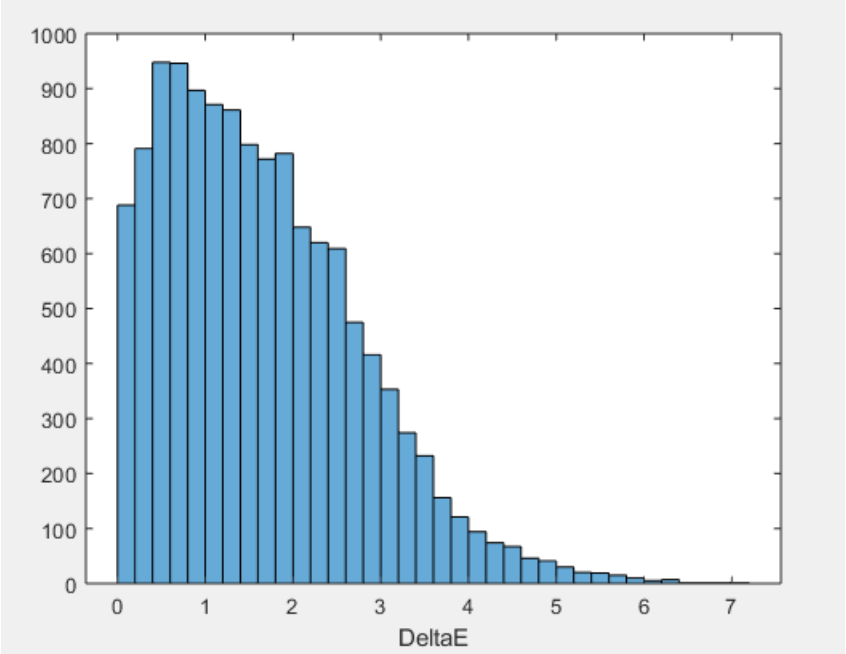

Figure 1.2: Istogramma delle DeltaE delle coppie (master, copia disturbata)

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 Training

Per effettuare il training della rete abbiamo usato la seguente funzione MATLab. Questa esegue 10 repetion ed al termine di ognuna stampa il relativo *Mean Squared Error* ed il coefficiente R della retta di regressione tra gli output della rete ed i relativi target.

```
function [best_mse, best_net] = fs_net_final(x,t)
    hiddenLayerSize = 10;
    repetitions = 10;
    best_mse = 999;
    disp(['fs_net_final started (', num2str(repetitions), ' repetitions
        )']);

    xx = x.';
    tt = t.';
    for i=1:repetitions
        net = fitnet(hiddenLayerSize);
        net.trainParam.showWindow = false;

        [net, tr] = train(net, xx, tt,'useParallel','yes');
        y = net(xx);
```

```
        perf = perform(net, tt, y);

        regr = regression(tt,y);
        disp(['Repetition ', num2str(i), ' mse=', num2str(perf), ' R=',
            num2str(regr)]);
        if(perf < best_mse)
            best_mse = perf;
            best_net = net;
        end
    end

    % plot best network regression
    [m,n] = size(x);
    y = best_net(xx);
    best_R = regression(tt,y);
    disp(['Best neural network (', num2str(n),' features ,', num2str(
        hiddenLayerSize), ' neurons): mse=', num2str(best_mse), ' R=',
        num2str(best_R)]);
    plotregression(tt,y, 'All');
end
```

1.4 Risultati

La rete che ha i risultati migliori è quella con 12 neuroni nello strato di ingresso e 10 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 skewness è calcolata dal mean e dalla varianza).

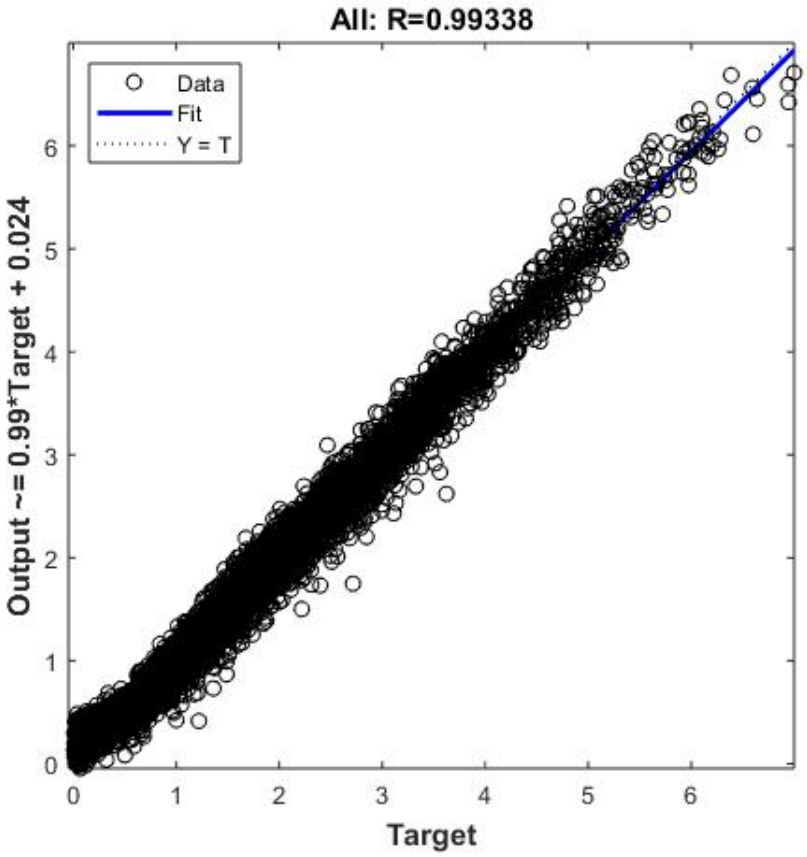

Figure 1.3: Retta di Regressione della rete neurale

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 Mamdani. 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.

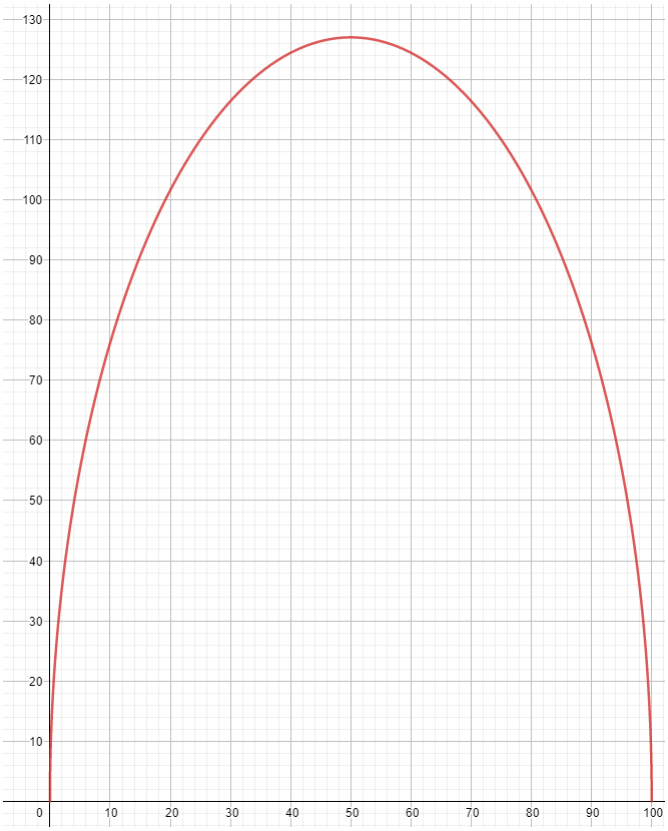

Figure 2.1: Grafico del C'_{max} al variare di L

Le funzioni di membership e le regole della rete fuzzy sono state progettate sulla base della percezione delle differenze delle coppie, analizzando varie aree dello spazio dei colori. Abbiamo notato che, a parità di DeltaE, la differenza delle coppie selezionate può essere percepita diversamente. Per questo motivo abbiamo individuato una serie di zone dello spazio LCh nella quale la percezione è differente da quella evidenziata dal DeltaE.

Colori Insaturi I colori poco saturi si distinguono con più difficoltà.

Colori Non Luminosi I colori poco luminosi, similmente a quelli insaturi, non sono facilmente distinguibili.

Zona colori Arancione I colori tendenti all'arancione sono più distinguibili.

Zona colori Giallo/Verde/Blu I colori tendenti al giallo, al verde o al blu non sono facilmente distinguibili.

2.1 Funzioni di Membership

Per identificare le zone elencate abbiamo modellato gli ingressi in base al:

- il colore (blue, yellow, green, ...)
- il livello di saturazione (low/high)
- il livello di luminosità (low/mid/high)
- il DeltaE dal punto di vista di un osservatore
- infine, per l'output, il DeltaE corretto, similmente al DeltaE in ingresso

Per quanto riguarda il DeltaE e il DeltaE Corretto ci siamo basati sulla differenza percepita dall'osservatore, identificando le seguenti funzioni di membership:

- $0 < \Delta E < 1$ **no-diff**: l'osservatore non percepisce differenze;
- $1 < \Delta E < 2$ **exp-diff**: solo osservatori con esperienza riescono a notare una differenza;
- $2 < \Delta E < 3.5$ **unexp-diff**: osservatore senza esperienza riescono a notare la differenza;
- $3.5 < \Delta E < 5$ **clear-diff**: i colori sono facilmente distinguibili;
- $\Delta E \geq 5$ **total-diff**: i colori sono distinti.

Le funzioni di membership effettive sono mostrate nella figura 2.2.

Per il colore (Hue) abbiamo utilizzato i range indicati nella tabella della figura 2.3[1].

[1] `https://www.researchgate.net/profile/Nele_Dael/publication/299491827/figure/fig6/AS:349601765838853@1460362964268/CIE-LCh-hue-categorization.png`

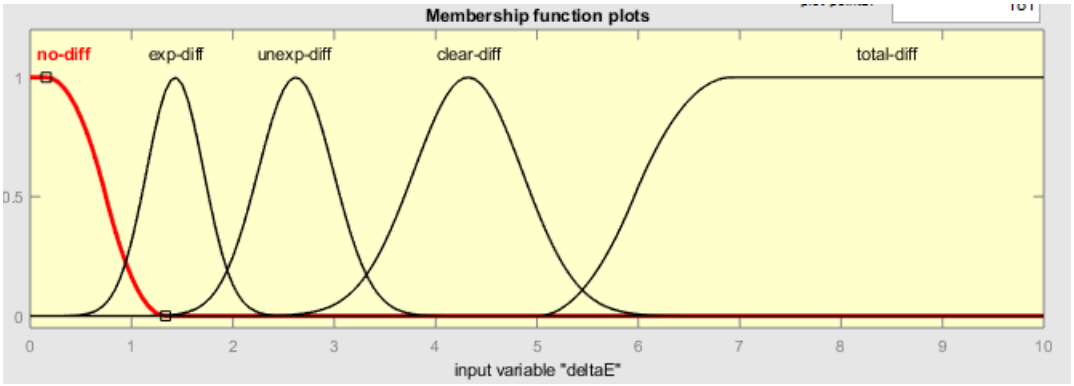

Figure 2.2: Funzione di membership DeltaE (distanza euclidea)

Colour	Focal hue	Hue range	Lightness
Red	25°	346°-40°	any
Orange	57°	40°-72°	any
Yellow	87°	72°-105°	any
Yellow-Green	116°	105°-130°	any
Green	144°	130°-166°	any
Green-Blue	194°	166°-220°	any
Blue	244°	220°-275°	any
Purple	306°	275°-346°	any
Achromatic	none	any	any

Figure 2.3: Tabella per la classificazione del parametro Hue

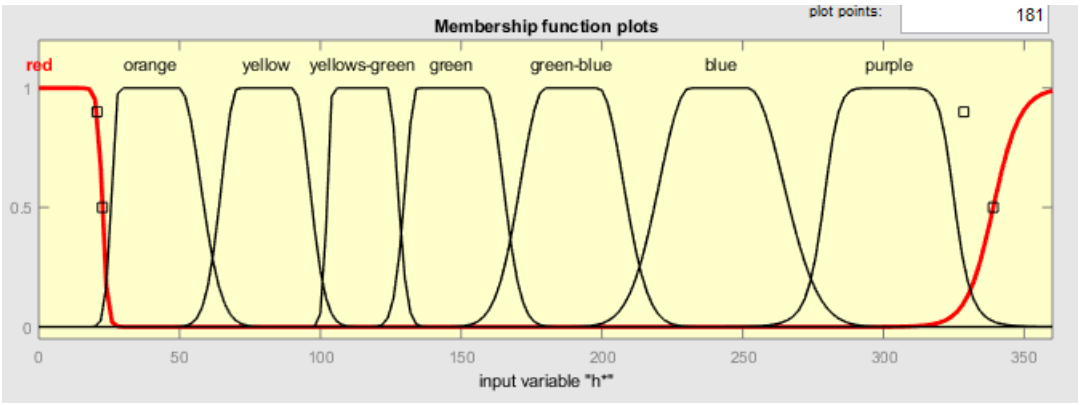

Figure 2.4: Funzione di membership Hue (Colore)

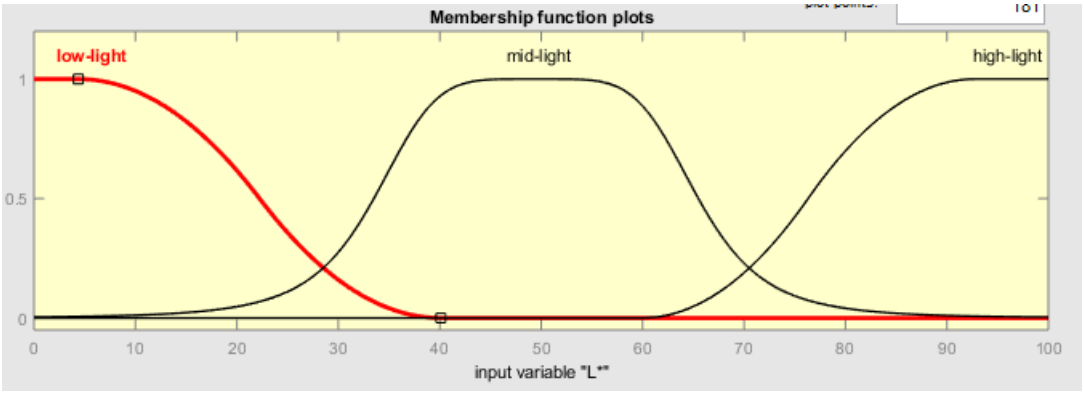

Figure 2.5: Funzione di membership Light

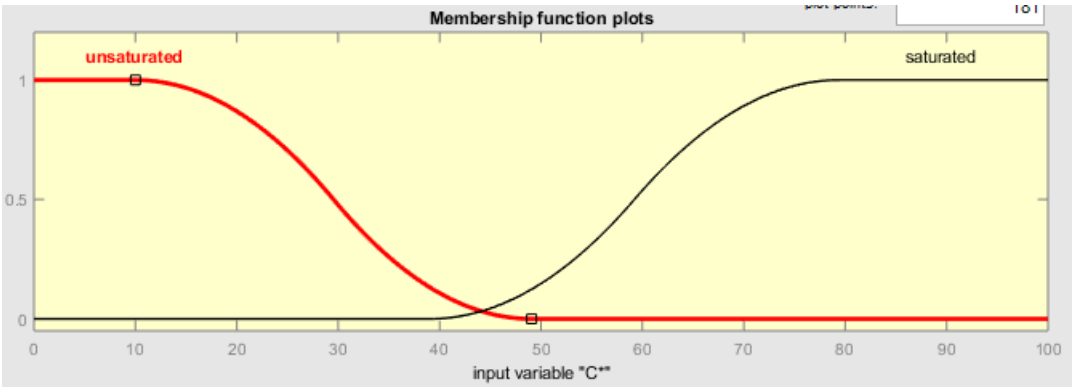

Figure 2.6: Funzione di membership Saturation

2.2 Regole

In base alle zone dello spazio LCh che abbiamo elencato prima, mostriamo le regole inserite nella rete Mamdani per la correzione dei DeltaE:

L	c	h	ΔE_{orig}	ΔE_{adj}	peso
-	-	-	total-diff	total-diff	0.2
-	-	-	clear-diff	clear-diff	0.2
-	-	-	unexp-diff	unexp-diff	0.2
-	-	-	exp-diff	exp-diff	0.2
-	-	-	no-diff	no-diff	0.2
low-light	-	-	total-diff	exp-diff	1
low-light	-	-	clear-diff	exp-diff	1
low-light	-	-	unexp-diff	no-diff	1
low-light	-	-	exp-diff	no-diff	1
mid-light	saturated	orange	unexp-diff	exp-diff	1
mid-light	saturated	green	unexp-diff	exp-diff	1
mid-light	saturated	green	exp-diff	no-diff	1
mid-light	saturated	blue	clear-diff	exp-diff	1
high-light	saturated	yellow	unexp-diff	no-diff	1
high-light	saturated	yellow	clear-diff	exp-diff	1
not(low-light)	unsaturated	-	not(no-diff)	exp-diff	1
not(low-light)	unsaturated	-	no-diff	no-diff	1

Table 2.1: Regole di inferenza del sistema Madamani

2.3 Risultati

Quello che ci aspettavamo, con l'implementazione della rete, era una differente distribuzione dei DeltaE corretti rispetto a quelli iniziali: questo risultato è confermabile guardando gli istogrammmi in figura 2.7. Le operazioni di fuzzyfication e defuzzyfication hanno concentrato, maggiormente, i DeltaE nelle zone specificate dalla funzione di membership e dalle regole di inferenza inserite.

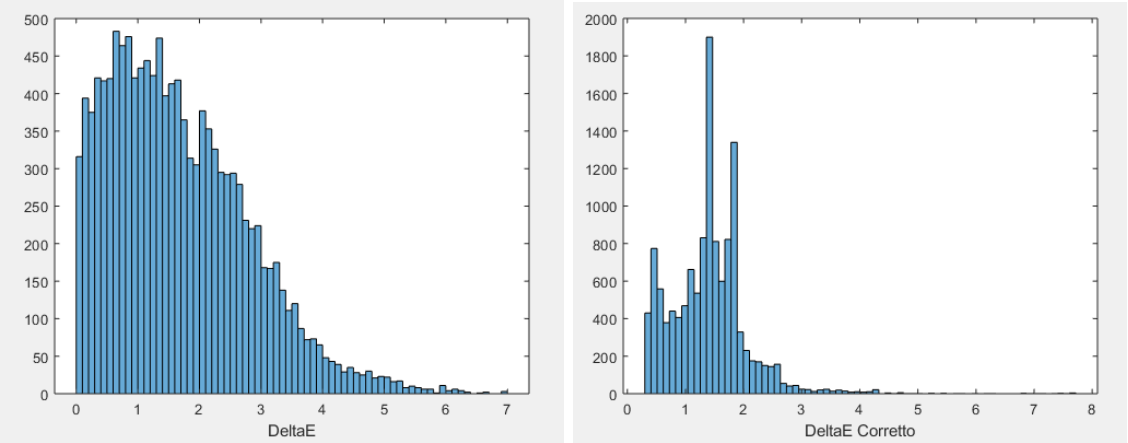

Figure 2.7: Comparazione Istogrammi DeltaE Originali / DeltaE Corretti

2.3.1 Risultati Rete Neurale

L'utilizzo dei DeltaE corretti per il training di una nuova rete neurale ha scaturito la necessità di estrarre e selezionare, nuovamente, le feature. In particolare le nuove feature estratte sono state:

master: var(1), median(1), median(3), skewness(5), maxYValue(2), indexAtMaxY(1)
copy: median(1), median(3), skewness(1), skewness(5), maxYValue(2), indexAtMaxY(2)

Estraendo 12 feature e utilizzando 10 neuroni nell'hidden layer (1 layer), abbiamo riscontrato:

$$Mean\ Squared\ Error \simeq 0.0395, R \simeq 0.954$$

Le performance della rete sono peggiorate, a parità di numero di feature e neuroni dello strato nascosto. Questo è imputabile al fatto che la rete fuzzy rende più complesso il legame tra i colori in ingresso e le distanze calcolate: nel primo caso la distanza era euclidea, nel secondo caso è modificata secondo le regole della rete fuzzy (quindi è una funzione più complessa per il fitting).

L'aumento del numero delle feature non aumenta in maniera significativa il valore dell'MSE o del coefficiente R, quindi consideriamo 12 feature (e 10 neuroni) come il miglior compromesso tra complessità e prestazioni della rete. Il Mean Squared Error risulta essere di tre ordini più piccolo rispetto al massimo valore di DeltaE previsto, cioè 10, quindi può essere considerato accettabile.

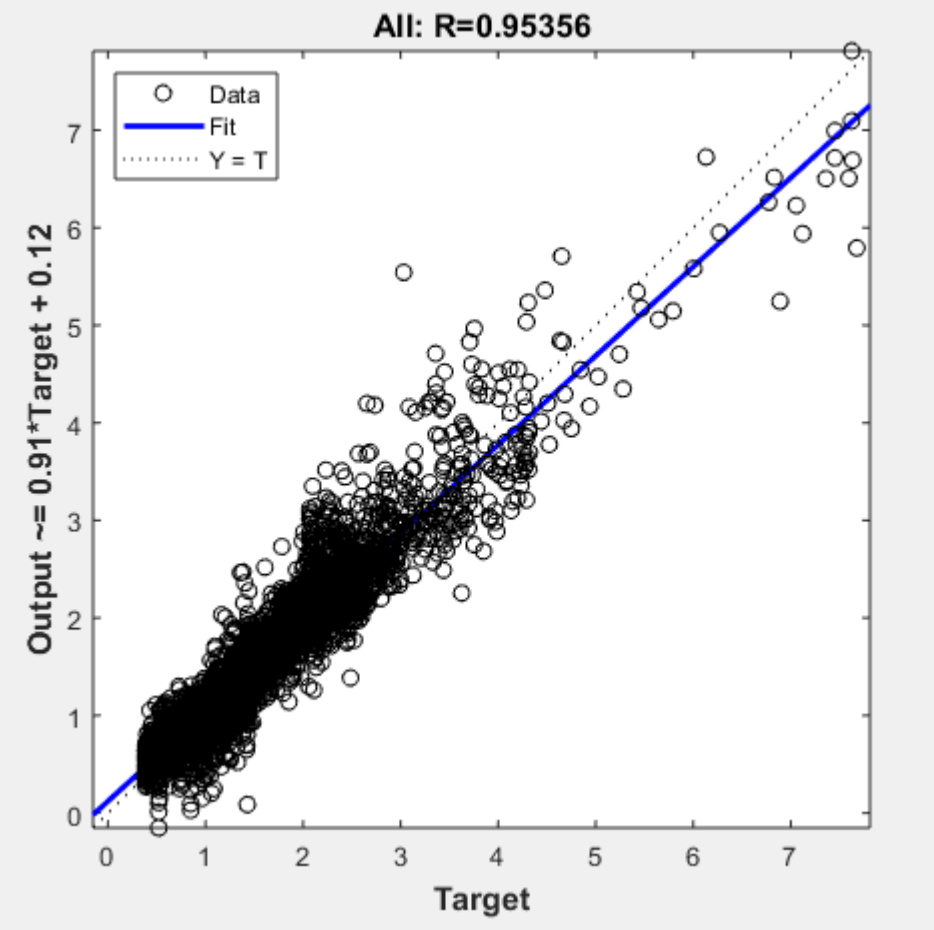

Figure 2.8: Retta di Regressione della rete neurale, dopo l'introduzione dei DeltaE corretti