MASTER DEGREE in EMBEDDED COMPUTING SYSTEMS

A.Y. 2017 - 2018

Relazione Progetto

Computational Intelligence

Professor.ssa
Beatrice Lazzerini

Studenti
Silvio Bacci
Andre Baldecchi

Indice

1 Introduzione

1.1 Introduzione

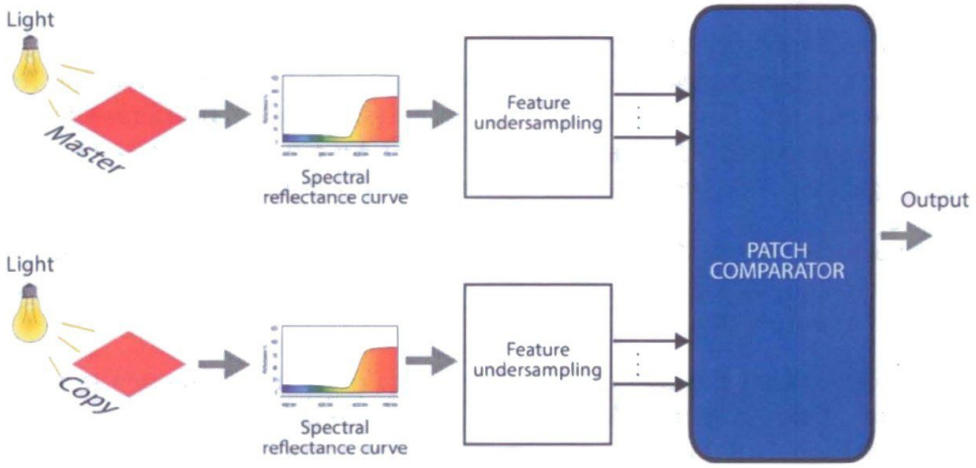
Come indicato dalle specifiche di progetto, l'obiettivo di questo lavoro è quello di sviluppare un sistema basato su rete neurale in grado di confrontare due colori e misurarne la loro similarità. Ci sono state fornite 1269 patch di colore opaco e per ognuna di esse i relativi spettri, coordinate XYZ, RGB e L*a*b*. Da sottolineare come in realtà siano state utilizzate soltanto 1232 delle 1269 totali in quanto 37 patch presentavano RGB negativi che i nostri calcolatori non erano in grado di rappresentare.

1.2 Obiettivo del progetto

L'obiettivo di questo progetto è quello di sviluppare un tool in grado di giudicare l'accuratezza con cui un processo di stampa industriale è in grado di riprodurre colori partendo da un determinato colore master. Questo tool potrebbe diventare di fondamentale importanza in ambito industriale visto che un requisito di qualità del processo di stampa implica che ogni riproduzione del master sia cromaticamente identica al master stesso. In particolar modo il tool (basato su rete neurale) che andiamo a proporre, mira a stimare la similarità cromatica soltanto tra coppie master-copia la cui differenza è quasi impercettibile ad occhio umano. Infatti, il processo industriale già prevede una fase dedicata alla valutazione oculare della stampa.

1.3 Struttura del tool

L'utilizzo tipico di un tool di questo tipo vedrà inizialmente una fase di campionamento con la quale, partendo da due superfici, due distribuzioni spettrali di potenza vengono ottenute. Queste rappresentano i colori di master e copia presi in considerazione. Come vedremo, il range di lunghezze d'onda visibili dall'occhio umano è ampio 420 nm. La rappresentazione su calcolatore di una distribuzione di questo tipo può, in alcuni casi, generare overhead dal punto di vista computazionale. Per questo, subito dopo il campionamento, sarà prevista una fase di undersampling con la quale la mole di dati da dover fornire alla rete neurale verrà ridotta. Le coppie di patch verranno a questo punto passate alla rete che, fornirà un output rappresentante il grado di similarità tra le patch in ingresso.



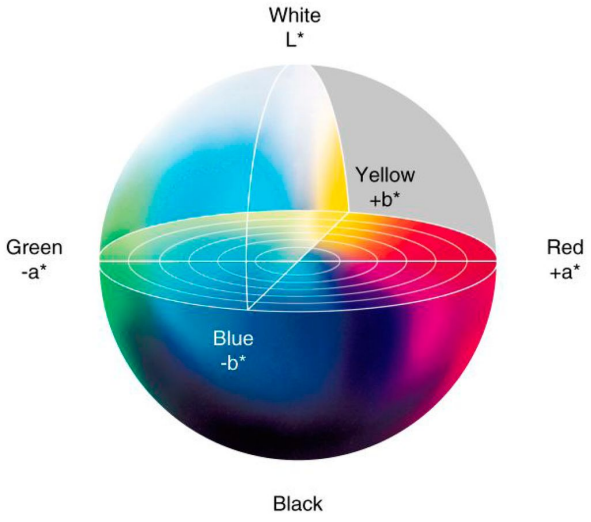
2 Spazio di colore L*a*b*

2.1 Introduzione

In questo capitolo andremo a descrivere lo spazio di colore L*a*b*, ponendo particolare attenzione ai modelli di calcolo relativi a distanze punto-punto nello spazio stesso.

2.2 Spazio di colori

Lo spazio colore L*a*b* o CIE L*a*b* o CIE 1976 (L*, a*, b*) è uno spazio colore-opponente con la dimensione L per la luminosità e a e b per i colori, basato sulle coordinate dello spazio di colore non lineare compresso CIE XYZ.



Una delle particolarità dello spazio L*a*b* risiede nella correlazione tra la differenza percepita tra due colori e la loro distanza nello spazio. Pertanto è possibile utilizzare la distanza L*a*b* per ottenere informazioni relative al livello di similarità di due colori.

2.3 Distanze

Il primo modello di calcolo adottato per la distanza L*a*b* che separa due colori è stato quello della distanza Euclidea (1976). Dati due colori (L_1^*, a_1^*, b_1^*) e (L_2^*, a_2^*, b_2^*), la distanza euclidea è definita come:

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

Dopo diversi studi sperimentali è risultato che una distanza di circa 2.3 rappresenta la minima differenza tra colori percepibile dall'occhio umano.

Tuttavia una caratteristica di questo spazio di colori è la presenza di non uniformità percettive al suo interno. In altre parole, considerando ad esempio due punti in una certa regione dello spazio aventi distanza pari a 1 e due punti in un'altra regione con la stessa distanza, le percezioni delle differenze di colori possono variare in base alle regioni considerate. Pertanto, la soglia sopracitata di 2.3 non rappresenta un valore costante su tutto lo spazio ma un valore che varia in funzione della regione. Per questo motivo abbiamo considerato un modello di calcolo differente.
Il modello di calcolo CIE 94 è stato introdotto nel 1994 per risolvere i problemi di non uniformità dello spazio L*a*b*. Riportiamo di seguito, i calcoli relativi a questo modello:

$$\Delta E^*_{94} = \sqrt{\left(\frac{\Delta L^*}{k_L S_L}\right)^2 + \left(\frac{\Delta C^*_{ab}}{k_C S_C}\right)^2 + \left(\frac{\Delta H^*_{ab}}{k_H S_H}\right)^2}$$

$$\Delta L^* = L^*_1 - L^*_2$$
$$C^*_1 = \sqrt{a^{*2}_1 + b^{*2}_1}$$
$$C^*_2 = \sqrt{a^{*2}_2 + b^{*2}_2}$$
$$\Delta C^*_{ab} = C^*_1 - C^*_2$$
$$\Delta H^*_{ab} = \sqrt{{\Delta E^*_{ab}}^2 - \Delta L^{*2} - {\Delta C^*_{ab}}^2} = \sqrt{\Delta a^{*2} + \Delta b^{*2} - {\Delta C^*_{ab}}^2}$$
$$\Delta a^* = a^*_1 - a^*_2$$
$$\Delta b^* = b^*_1 - b^*_2$$
$$S_L = 1$$
$$S_C = 1 + K_1 C^*_1$$
$$S_H = 1 + K_2 C^*_1$$

Nel 2000 un ulteriore modello è stato proposto (CIEDE2000). Questo aggiunge 5 correzioni sulla non uniformità al modello precedente (CIE94). Abbiamo scelto di adottare questo modello in quanto ad oggi risulta essere il più preciso. Come per il caso precedente riportiamo di seguito i calcoli relativi a questo modello.

$$\Delta E_{00}^{*} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \frac{\Delta C'}{k_C S_C}\frac{\Delta H'}{k_H S_H}}$$

$$\Delta L' = L_2^* - L_1^*$$
$$\bar{L} = \frac{L_1^* + L_2^*}{2} \quad \bar{C} = \frac{C_1^* + C_2^*}{2}$$
$$a_1' = a_1^* + \frac{a_1^*}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right) \quad a_2' = a_2^* + \frac{a_2^*}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)$$
$$\bar{C}' = \frac{C_1' + C_2'}{2} \text{ and } \Delta C' = C_2' - C_1' \quad \text{where } C_1' = \sqrt{a_1'^2 + b_1^{*2}} \quad C_2' = \sqrt{a_2'^2 + b_2^{*2}}$$
$$h_1' = \operatorname{atan2}(b_1^*, a_1') \mod 360^\circ, \quad h_2' = \operatorname{atan2}(b_2^*, a_2') \mod 360^\circ$$

$$\Delta h' = \begin{cases} h_2' - h_1' & |h_1' - h_2'| \le 180^\circ \\ h_2' - h_1' + 360^\circ & |h_1' - h_2'| > 180^\circ, h_2' \le h_1' \\ h_2' - h_1' - 360^\circ & |h_1' - h_2'| > 180^\circ, h_2' > h_1' \end{cases}$$

$$\Delta H' = 2\sqrt{C_1' C_2'}\sin(\Delta h'/2), \quad \bar{H}' = \begin{cases} (h_1' + h_2' + 360^\circ)/2 & |h_1' - h_2'| > 180^\circ \\ (h_1' + h_2')/2 & |h_1' - h_2'| \le 180^\circ \end{cases}$$

$$T = 1 - 0.17\cos(\bar{H}' - 30^\circ) + 0.24\cos(2\bar{H}') + 0.32\cos(3\bar{H}' + 6^\circ) - 0.20\cos(4\bar{H}' - 63^\circ)$$
$$S_L = 1 + \frac{0.015\left(\bar{L} - 50\right)^2}{\sqrt{20 + \left(\bar{L} - 50\right)^2}} \quad S_C = 1 + 0.045\bar{C}' \quad S_H = 1 + 0.015\bar{C}'T$$
$$R_T = -2\sqrt{\frac{\bar{C}'^7}{\bar{C}'^7 + 25^7}}\sin\left[60^\circ \cdot \exp\left(-\left[\frac{\bar{H}' - 275^\circ}{25^\circ}\right]^2\right)\right]$$

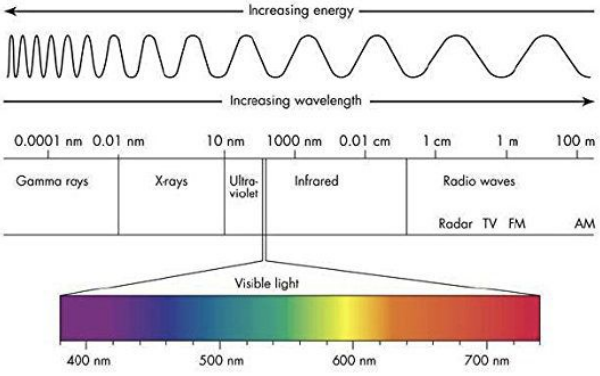
3 Generazione copie

3.1 Introduzione

Considerando il fatto che un colore può essere rappresentato attraverso una Spectral Power Distribution (SPD), possiamo trattare ogni colore come un generico segnale espresso dalla SPD stessa. Così facendo, la generazione delle copie avverrà attraverso l'applicazione di un rumore Gaussiano bianco al segnale di riferimento. In questo capitolo andremo inizialmente a descrivere e motivare i criteri con cui sono state generate le copie. Dopo di ché scenderemo più in dettaglio mostrando alcuni frammenti di codice Matlab utilizzati per generare le copie automaticamente.

3.2 Natura delle copie

Osserviamo con attenzione la seguente figura in cui vengono associate le lunghezze d'onda visibili dall'occhio umano ai colori:



Come si può notare, l'intervallo di lunghezze d'onda visibili dall'occhio umano va da 380 nanometri a 800 nanometri circa. Una prima possibilità è quella di generare copie applicando un rumore gaussiano bianco su intervalli di lunghezze d'onda limitati. In particolar modo, osservando la precedente figura, si nota che una sfumatore di colore è definita in un intervallo di lunghezze d'onda di circa 40 nm. Per questo motivo inizialmente abbiamo individuato 10 intervalli da 42 nm e abbiamo applicato il rumore singolarmente, generando cosi 10 copie per ogni master. Tuttavia, le copie generate in questo modo risultavano eccessivamente simili tra loro, per questo motivo abbiamo tentato di far variare anche l'intensità dei rumori applicati sui singoli intervalli ottenendo però un numero di copie per master elevato. Inoltre le differenze tra le diverse copie venivano sostanzialmente individuate dalla differente intensità del disturbo applicato. Per questo le copie, effettivamente utilizzate, sono state generate applicando un rumore gaussiano bianco distribuito sull'intero intervallo di lunghezze d'onda. Abbiamo scelto di far variare l'intensità del rumore applicato. In particolar modo abbiamo definito 4 diversi livelli di intensità, ottenendo quindi 4 copie per ogni master selezionato.

3.3 Valutazione e classificazione delle copie

Abbiamo scelto di classificare la differenza tra una coppia di colori utilizzando 3 classi di similarità. La prima classe contiene quindi coppie di patch indistinguibili tra loro ad occhio, la seconda contiene coppie di colori la cui differenza è appena percettibile mentre l'ultima classe rappresenta coppie totalmente differenti. Al fine di produrre i target per la nostra rete abbiamo valutato soggettivamente un insieme di coppie di colori appartenenti a regioni dello spazio L*a*b* differenti. Questa valutazione ha messo in luce diverse problematiche. Prima di tutto, la valutazione è differente da individuo ad individuo (non sono state poche le situazioni di incongruenza tra i membri del gruppo). Inoltre valutazioni successive delle stesse coppie di colori da parte dello stesso individuo sono risultate sostanzialmente differenti. In aggiunta, la valutazione di un individuo è facilmente suggestionabile da informazioni ricevute prima della valutazione stessa (ad esempio, la conoscenza del relativo ΔE). Infine, abbiamo notato che la percezione dei colori e delle loro differenze dipende fortemente anche dal tipo di display adottato.

3.4 Implementazione software

La funzione Matlab responsabile della generazione di una copia è la seguente.

[spectra, rSPD, coordinate] = create_copy(master_spectra, db, wl_noise, db2, wl_noise2)

Questa funzione permette di creare una copia applicando rumore su due diversi intervalli di lunghezze d'onda. I 4 parametri:

- Master dal quale creare la copia
- Intensità del segnale di disturbo da applicare al primo intervallo
- Primo intervallo da disturbare
- Intensità del segnale di disturbo da applicare al secondo intervallo
- Secondo intervallo da disturbare

Nel caso in cui l'ultima coppia di parametri non venga specificata, la funzione si limita ad applicare rumore su un solo intervallo di lunghezze d'onda. I valori di ritorno di questa funzione non sono altro che spettro, curva di riflettanza e coordinate RGB, XYZ e L*a*b* della copia creata.
Il disturbo viene applicato attraverso l'ausilio delle funzione *awgn* di Matlab. Questa prende come parametri la porzione di segnale da disturbare e l'intensità in dB del disturbo. Tale intensità viene espressa come rapporto fra la potenza del segnale e quella del disturbo (SNR). Il secondo parametro della funzione *awgn* sarà quindi calcolato come:

$$10 \cdot log_{10}(\tfrac{S}{N})$$

Ovviamente l'intensità del rumore desiderato, sarà inferiore a quella segnale da disturbare. In particolare all'aumentare dei dB corrisponderà una diminuzione dell'intensità del disturbo. Per questo motivo, dopo un certo numero di prove, abbiamo individuato 4 diverse intensità di rumore: 15, 25, 35 e 45 dB.

Con queste intensità, il disturbo rispetto al segnale d'origine varierà tra i 2 valori riportati di seguito:

$$10 \cdot log_{10}(\tfrac{S}{N}) = 15 \quad \rightarrow \quad \tfrac{S}{N} \approx 30$$

$$10 \cdot log_{10}(\tfrac{S}{N}) = 45 \quad \rightarrow \quad \tfrac{S}{N} \approx 3 \cdot 10^{4}$$

Pertanto l'intensità del disturbo varierà tra 1/30 e un 1/30000 del segnale originale. Con i 4 valori di intensità scelti sperimentalmente le copie generate saranno distribuite tra le 3 classi di similarità definite in modo abbastanza omogeneo. Questa scelta deriva dal fatto di voler evitare che la nostra rete basi le sue decisioni su un sola tipologia di input. La distribuzione omogenea dei campioni infatti, assicura generalmente delle migliori prestazioni della rete finale.

4 Definizione degli insiemi di training, test e validation

4.1 Introduzione

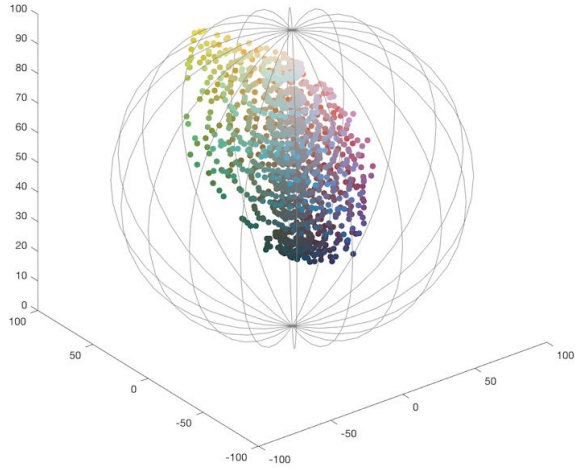
Per ogni master selezionato vengono generate 4 copie con cui allenare la rete. Ogni coppia master-copia andrà singolarmente classificata specificando l'appartenenza ad una delle 3 classi di similarità. Questo compito risulta essere molto oneroso in quanto il dataset di partenza comprende 1232 master. Il numero totale di valutazioni da fare sarebbe circa 5000. Abbiamo quindi scelto di non utilizzare tutto il dataset a nostra disposizione ma solamente il 40%. La selezione dei master avviene mediante l'uso di un algoritmo di clustering.

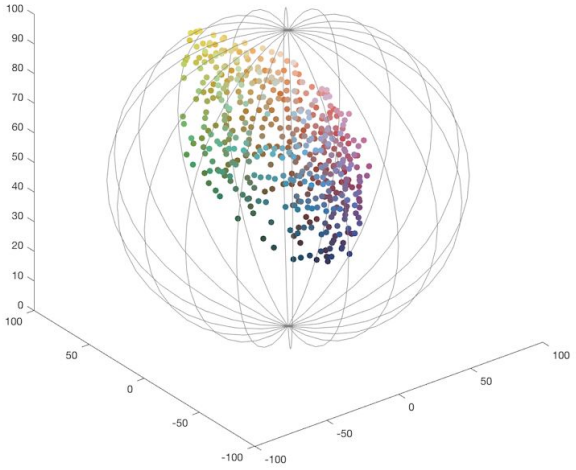
4.2 Selezione dei master

La procedura di selezione inizia con il calcolo delle distanze tra tutti i master del dataset nello spazio L*a*b*. Tali distanze vengono memorizzate in una matrice simmetrica 1232x1232 in cui, la cella (i,j) denota la distanza tra i master di indice i e j. Queste distanze vengono quindi sfruttate per andare a filtrare i master eccessivamente simili. La procedura di filtraggio è composta dai seguenti passi:

1. Ricerca del minimo elemento (i,j) della matrice
   1.1. Il contenuto della cella (i,j) viene settato ad infinito
2. Eliminazione dell'elemento di indice i dal dataset
3. Incremento del numero di elementi eliminati
4. Se il numero degli elementi eliminati è inferiore a 739 (60% di 1232) saltare al passo 1.

I master selezionati sono così rappresentati nello spazio L*a*b*:





4.3 Clustering

A questo punto rimangono da definire i 3 set su cui allenare la rete:

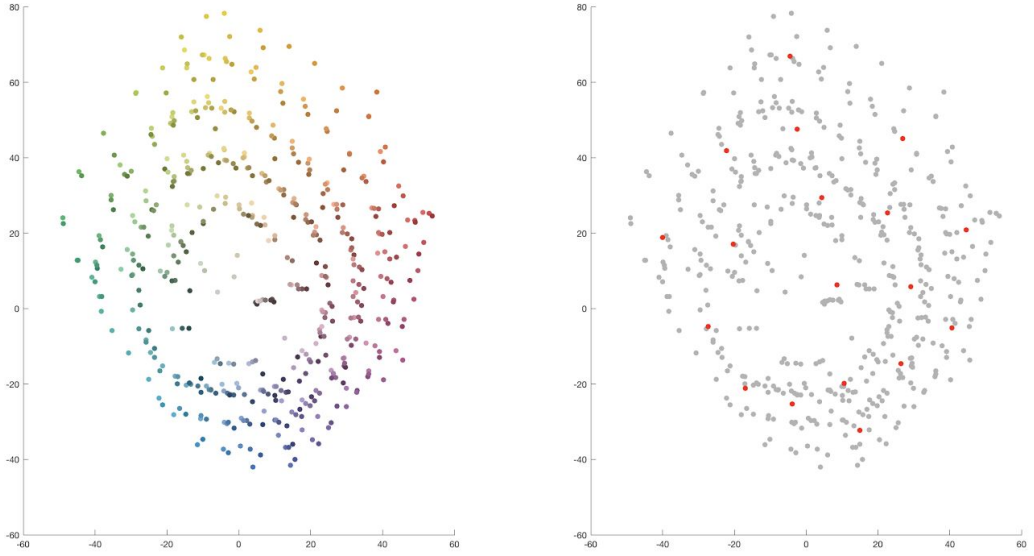
- Training set
- Test set
- Validation set

La definizione di questi 3 set è un punto cruciale per la creazione di una rete neurale che riesca ad apprendere correttamente il problema. Ognuno di questi insiemi deve rappresentare in modo indipendente il problema che la rete deve affrontare. Considerando questo, abbiamo cercato di includere in ognuno dei 3 set menzionati le principali sfumature di colori presenti nel dataset. In questo modo l'allenamento, il testing e la validazione della rete avverranno in funzione dello stesso set di sfumature cromatiche. Per esempio, evitiamo di allenare la rete soltanto con sfumature di colore blu e testarla con sfumature di colore verde.
Inoltre un'importante caratteristica di questi 3 insiemi è la completezza, ovvero coprire tutte le possibili sfumature di colore. Per risolvere questo problema abbiamo eseguito un clustering del dataset in funzione della distanza L*a*b* tra i diversi master. Abbiamo utilizzato inizialmente due metodi, clustering mediante l'utilizzo dell'algoritmo iterativo *k-means* e mediante l'utilizzo di una self organized map (SOM) propriamente allenata. Entrambi richiedono inizialmente il numero di cluster da definire. Per questo, abbiamo valutato soggettivamente il dataset a nostra disposizione individuando indicativamente 13 sfumature di colore. Abbiamo quindi deciso di provare entrambi gli approcci su un numero di cluster compreso tra 10 e 20. Il clustering prodotto dalla SOM è risultato in ogni caso peggiore dei clustering prodotti da *k-means.*

L'algoritmo di clustering utilizzato è quindi *k-means.* Il numero ottimale di cluster viene individuato calcolando e valutando l'indice *Xie-Beni.* Questo indice è derivato dal quoziente tra due espressioni rappresentanti compattezza e separazione dei cluster, rispettivamente numeratore e denominatore del quoziente.

$$\frac{\sum_{i=1}^{k}\sum_{j=1}^{N} u_{ik}{}^{m} \| x_j - v_i \|^2}{N \cdot min_{i \neq k}(\| v_i - v_k \|)^2}$$

Per questo motivo, minore è l'indice migliore è il clustering associato. Nel nostro caso il numero ottimale di cluster è risultato 18, valore indicativamente in linea con la nostra valutazione soggettiva.
Al termine di questa fase, la porzione del dataset considerata viene divisa in k cluster, ognuno contenente master appartenenti la stessa sfumatura di colore.



4.4 Insiemi di training, testing e validation

I master inclusi in ogni cluster vengono quindi suddivisi nei tre set in modo casuale. In particolare il 70% dei master viene incluso nell'insieme di training, il restante 30% viene equamente distribuito sui due insiemi rimanenti, test e validation.
In questo modo, ogni insieme, conterrà un insieme di master rappresentante l'intero dataset di partenza. I tre insiemi conterranno inoltre le stesse sfumature di colori. Abbiamo quindi definito la nostra rete utilizzando un numero totale di campioni pari a 1968 ($4 \cdot 492$) di cui, 1452 (70%) utilizzati per allenare la rete, 256 per testarla e le restanti 252 per l'insieme di validazione.

5 Reti Neurali

5.1 Introduzione

In questo capitolo andremo a descrivere le reti neurali prodotte, commentando successivamente le loro differenze. In particolar modo abbiamo creato Multilayer perceptron network e Radial basis function network.

5.2 Multilayer perceptron network (MLP)

MLP è un modello di rete neurale artificiale che mappa insiemi di dati in ingresso in un insieme di dati in uscita appropriati. La *neural network toolbox* di Matlab offre diversi tipi di reti MLP, le due da noi adottate sono *patternet* e *fitnet*. Le reti fitnet, create dall'omonima funzione Matlab, sono reti MLP che offrono particolari prestazioni per quanto riguarda il problema di function fitting. Le reti patternet invece, sono reti ottimizzate per la risoluzione di problemi di classificazione e pattern recognition. Entrambe ammettono come parametri:

- Numero di neuroni di ciascun hidden layer.
- Algoritmo di training

L'algoritmo di training utilizzato è quello di default ovvero *scaled conjugate gradient backpropagation*, un metodo iterativo a retropropagazione basato sulla valutazione del gradiente della funzione obiettivo. L'unico parametro da far variare è quindi il numero di neuroni di ciascuno hidden layer.

Una prima soluzione prevede un doppio ciclo innestato in cui, esternamente, si fa variare il numero di neuroni mentre internamente, la rete viene allenata 10 volte utilizzando l'insieme di training. Alla fine di ogni ciclo di training viene calcolata l'accuratezza che indica il rate con cui la rete risponde in modo corretto.

Una seconda soluzione è quella di introdurre un ulteriore hidden layer. Oltre che il numero di neuroni del primo hidden layer, anche il numero di neuroni del secondo varierà. Per questo, si individueranno 3 cicli innestati. Il più esterno scorrerà il numero di neuroni del primo hidden layer, il secondo scorrerà il numero di neuroni del secondo hidden layer mentre il terzo, come nel caso precedente, sarà relativo al numero di allenamenti.

Entrambe le soluzioni appena trattate, sono state implementate utilizzando sia la funzione *fitnet* che la funzione *patternet* ottenendo così 4 reti di tipo MLP.

5.3 Radial Basis Function network (RBF)

RBF è un modello di rete neurale artificiale che usa funzioni di base radiale come funzioni di attivazione. Per questo modello di rete neurale abbiamo creato tre reti differenti.

Una prima soluzione prevede l'utilizzo della funzione *newrb*. Questa funzione permette la creazione di una rete RBF passando i seguenti parametri:

- Input vectors
- Target vectors
- Goal: valore di MSE per il quale terminare la costruzione della rete
- Spread
- Massimo numero di neuroni

Utilizziamo un doppio ciclo innestato in cui si fanno variare, esternamente, il numero di neuroni ed internamente lo spread. L'intervallo su cui far variare lo spread viene individuato calcolando la minima distanza tra vettori in input adiacenti e la massima distanza tra due qualsiasi vettori in input. Questi due estremi vengono quindi normalizzati dividendoli per la radice quadrata del numero di vettori considerati.
La selezione del numero di neuroni con cui creare la rete viene fatta modificando il parametro che indica il massimo numero di neuroni e il goal (settato pari a 0). Poiché arrivare ad un goal pari a 0 è molto improbabile, la *newrb* arriverà ad utilizzare il massimo numero di neuroni.

Una seconda soluzione prevede di definire centri e spread delle funzioni radiali in modo analogo a quello adottato nell'algoritmo di learning "Fixed centers selected at random". In questa soluzione quindi non si utilizza più la funzione *newrb* sia perchè il numero di neuroni effettivamente utilizzato potrebbe non essere quello indicato (se il goal viene raggiunto) sia perché la *newrb* non offre la possibilità di selezionare i centri ad hoc. Per questo motivo si è fatto uso della funzione Matlab *newrbe* che permette di scegliere il numero di neuroni (pari al numero di input che vengono forniti) ed i centri (esattamente nei punti indicati dagli input). Per la risoluzione di questa rete abbiamo scelto di far variare il numero di neuroni e allo stesso tempo trovare in modo casuale un sottoinsieme di ingressi (che saranno i centri) pari al numero di neuroni desiderati. Una volta calcolati i centri, l'intervallo dello spread viene calcolato in modo analogo al caso precedente, con l'unica differenza che per questa rete esso non varia, rimane costante e pari al estremo superiore dell'intervallo calcolato.

Una terza soluzione potrebbe essere molto simile alla precedente. L'unica differenza infatti risiede nell'algoritmo utilizzato per scegliere i centri. In questo infatti centri e spread vengono calcolati in modo analogo a quanto fatto per l'algoritmo di learning "Self organized selection centers". I centri quindi non vengono più scelti casualmente, ma come centroidi di un numero di cluster pari al numero di neuroni desiderati. In particolar modo, l'algoritmo utilizzato per la creazione dei cluster è stato *k-means*. Il problema sorto in questo caso, risiede nella natura del centroide stesso che, tipicamente, non coincide con uno dei punti del set di ingresso ma è un punto di "accumulazione". Per questo motivo la scelta adottata è stata quella di utilizzare come centro, l'input più vicino al centroide. Successivamente si calcola lo spread per normalizzazione come nel caso precedente.

6 Metodi per la valutazione delle performance

6.1 Introduzione

Per comparare le performance relative alle diverse reti create abbiamo considerato due alternative Il primo, quello visto fin'ora, seleziona la migliore rete in base a MSE e accuratezza. Il secondo invece fa uso di un sistema fuzzy MISO.

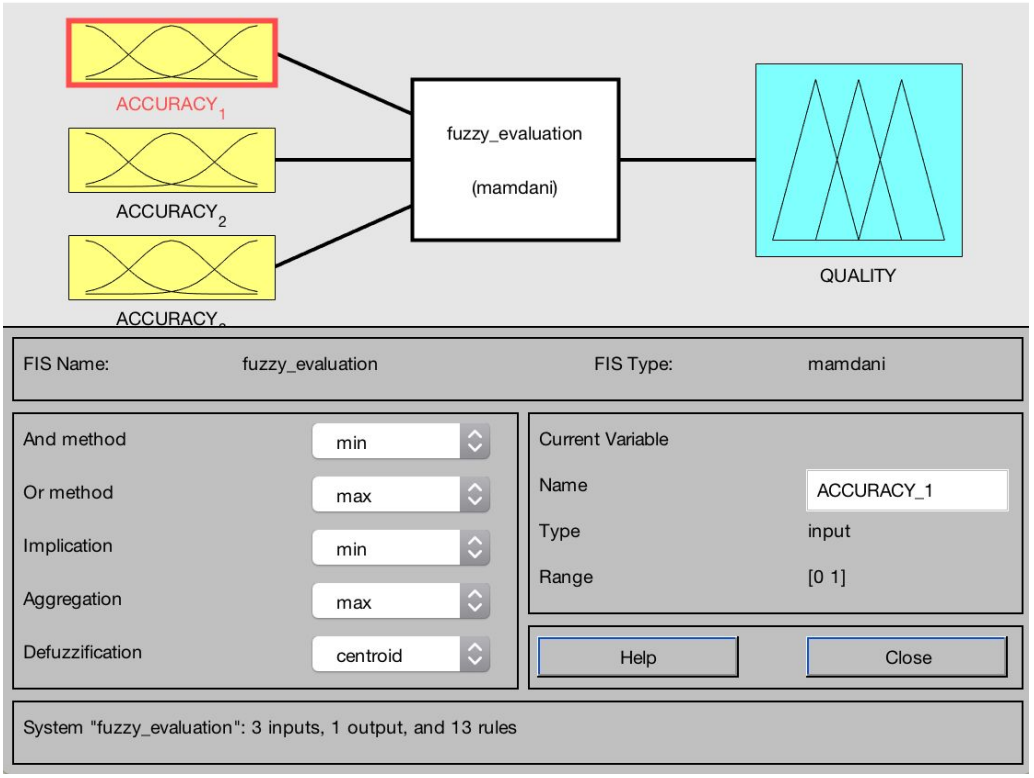
6.2 MSE e accuratezza

La prima soluzione è stata quella di valutare le performance delle diverse reti vengono confrontando MSE e accuratezza. Questi parametri vengono utilizzati per poter scegliere il numero ideale di neuroni con cui costruire la rete. Infatti si preferirà la soluzione con MSE minimo e accuratezza massima. Solitamente MSE e accuratezza sono 2 parametri correlati. Per questo il caso ottimale, che si è verificato piuttosto frequentemente, è quello in cui la stessa rete che contemporaneamente minimizza MSE e massimizza l'accuratezza. Nel caso in cui questo non si verifichi, la scelta adottata è stata quella di utilizzare la rete con accuratezza maggiore.

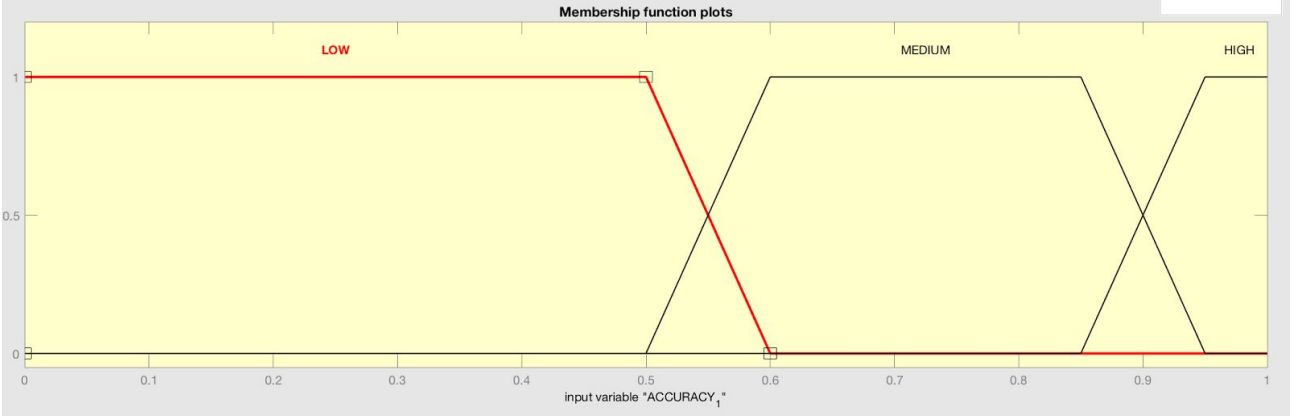
6.3 Sistema fuzzy

Un metodo migliore per la valutazione delle qualità delle reti viene individuato dall'utilizzo della logica fuzzy. E' possibile infatti definire un sistema fuzzy che, a partire dalla accuratezza delle singole classi, valuti la qualità della rete. Il sistema fuzzy sarà quindi di tipo MISO (multiple input-single output), avendo in ingresso le tre percentuali di accuratezza prodotte dalla rete associate alle classi di similarità. L'utilizzo del sistema fuzzy ci da la possibilità di assegnare diversi pesi alle accuratezze delle tre classi. Questo viene sostanzialmente realizzato definendo le regole su cui il sistema è basato. In particolare nel nostro caso abbiamo voluto dare una maggiore importanza alla classe 2 rispetto alle altre. Questo perchè, essendo la classe 2 la classe mediana, questa sarà la classe che risentirà di più dei difetti della rete. I casi in cui la classificazione risulta scorretta sono infatti individuati tra classe 1 e classe 2 e classe 2 e classe 3 (non tra le classi 1 e 3).

Il sistema fuzzy realizzato è strutturato nel seguente modo:

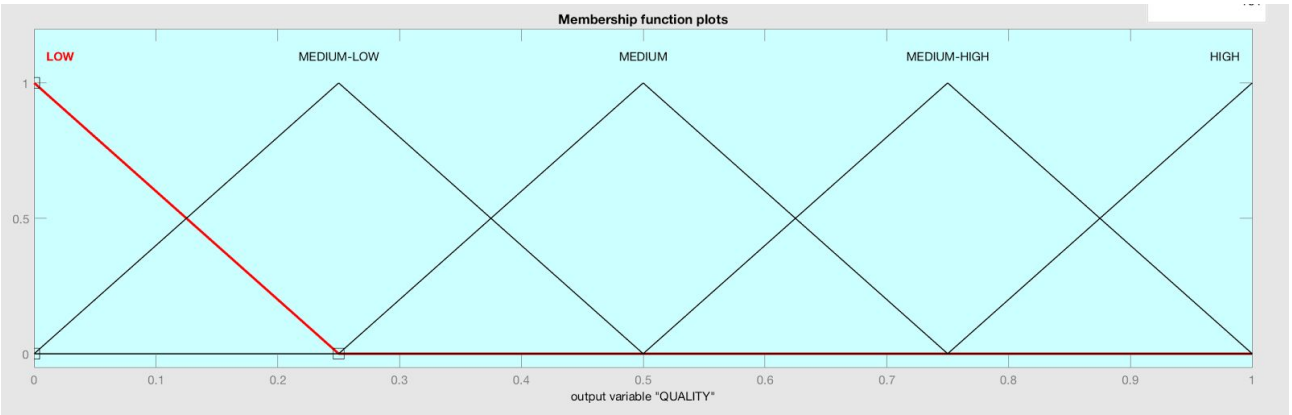


Iniziamo definendo le funzioni di appartenenza, per quanto riguarda i tre input:



Come si evince dai grafici delle funzioni, una percentuale di accuratezza viene classificata in 3 livelli, bassa, media ed alta.

La funzione d’appartenenza associata all’output sarà invece:



In questo caso invece, i livelli di valutazione dell’output sono 5, basso, medio-basso, medio, medio-alto e alto.

Per la definizione delle regole Mamdani, su cui si basa il sistema, abbiamo inizialmente prodotto la seguente tabella di verità.

ACCURACY 1	ACCURACY 2	ACCURACY 3	QUALITY
LOW	LOW	LOW	LOW
LOW	LOW	MEDIUM	LOW
LOW	LOW	HIGH	LOW
LOW	MEDIUM	LOW	MEDIUM-LOW
LOW	MEDIUM	MEDIUM	MEDIUM
LOW	MEDIUM	HIGH	MEDIUM
LOW	HIGH	LOW	MEDIUM
LOW	HIGH	MEDIUM	MEDIUM-HIGH
LOW	HIGH	HIGH	MEDIUM-HIGH
MEDIUM	LOW	LOW	LOW
MEDIUM	LOW	MEDIUM	MEDIUM-LOW
MEDIUM	LOW	HIGH	MEDIUM-LOW
MEDIUM	MEDIUM	LOW	MEDIUM
MEDIUM	MEDIUM	MEDIUM	MEDIUM
MEDIUM	MEDIUM	HIGH	MEDIUM-HIGH
MEDIUM	HIGH	LOW	MEDIUM-HIGH
MEDIUM	HIGH	MEDIUM	MEDIUM-HIGH
MEDIUM	HIGH	HIGH	HIGH
HIGH	LOW	LOW	LOW
HIGH	LOW	MEDIUM	MEDIUM-LOW
HIGH	LOW	HIGH	MEDIUM-LOW
HIGH	MEDIUM	LOW	MEDIUM
HIGH	MEDIUM	MEDIUM	MEDIUM-HIGH
HIGH	MEDIUM	HIGH	MEDIUM-HIGH
HIGH	HIGH	LOW	MEDIUM-HIGH
HIGH	HIGH	MEDIUM	HIGH
HIGH	HIGH	HIGH	HIGH

Questa tabella di verità, come si nota, associa ogni combinazione dei tre livelli di accuratezza in input ad un livello di qualità delle rete (output del sistema). Abbiamo poi riassunto le regole derivate dalla tabella specificandole nel formato delle regole Mamdani.

1. If (ACCURACY_1 is LOW) and (ACCURACY_2 is LOW) then (QUALITY is LOW) (1)
2. If (ACCURACY_2 is LOW) and (ACCURACY_3 is LOW) then (QUALITY is LOW) (1)
3. If (ACCURACY_1 is not LOW) and (ACCURACY_2 is LOW) and (ACCURACY_3 is not LOW) then (QUALITY is MEDIUM-LOW) (1)
4. If (ACCURACY_1 is LOW) and (ACCURACY_2 is MEDIUM) and (ACCURACY_3 is not LOW) then (QUALITY is MEDIUM) (1)
5. If (ACCURACY_1 is not LOW) and (ACCURACY_2 is MEDIUM) and (ACCURACY_3 is LOW) then (QUALITY is MEDIUM) (1)
6. If (ACCURACY_1 is MEDIUM) and (ACCURACY_2 is MEDIUM) and (ACCURACY_3 is MEDIUM) then (QUALITY is MEDIUM) (1)
7. If (ACCURACY_1 is not LOW) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is LOW) then (QUALITY is MEDIUM-HIGH) (1)
8. If (ACCURACY_1 is LOW) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is not LOW) then (QUALITY is MEDIUM-HIGH) (1)
9. If (ACCURACY_1 is not LOW) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is not LOW) then (QUALITY is MEDIUM-HIGH) (1)
10. If (ACCURACY_1 is HIGH) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is not LOW) then (QUALITY is HIGH) (1)
11. If (ACCURACY_1 is not LOW) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is HIGH) then (QUALITY is HIGH) (1)
12. If (ACCURACY_1 is MEDIUM) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is MEDIUM) then (QUALITY is HIGH) (1)
13. If (ACCURACY_1 is LOW) and (ACCURACY_2 is HIGH) and (ACCURACY_3 is LOW) then (QUALITY is MEDIUM-LOW) (1)
14. If (ACCURACY_1 is LOW) and (ACCURACY_2 is MEDIUM) and (ACCURACY_3 is LOW) then (QUALITY is MEDIUM-LOW) (1)

E’ importante notare che, se il sistema fuzzy individua due reti con la stessa qualità, la migliore verrà selezionata in base all’accuratezza della seconda classe che, come già detto, è la classe di maggior rilievo.

7 Valutazione delle performances delle reti

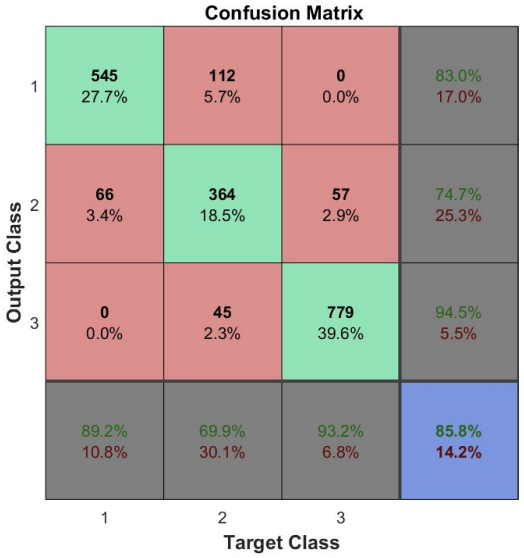
7.1 Introduzione

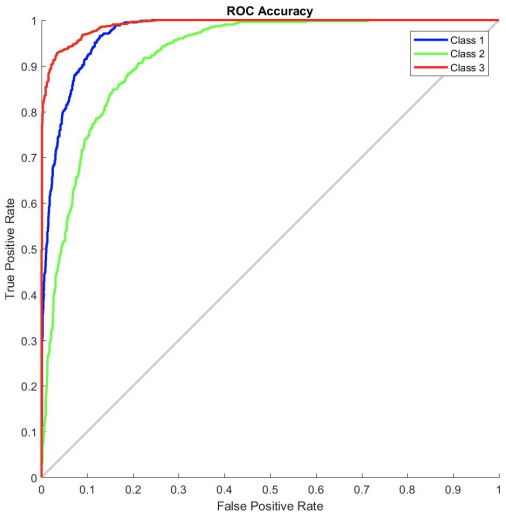
In questo capitolo andremo ad analizzare le effettive performance delle rete neurali prodotte. Per ognuna di queste, forniremo confusion matrix e curve ROC individuando infine la rete con la prestazioni migliori.

7.2 Performance MLP

7.2.1 Reti patternet

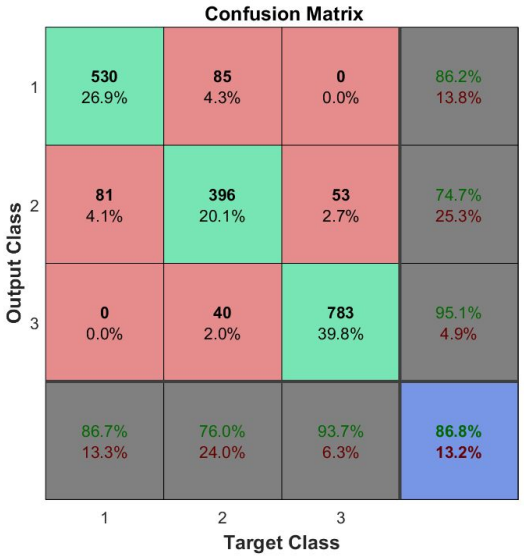
Le due figure seguenti, contengono le prestazioni della migliore rete patternet avente un solo hidden layer. In particolare, la rete selezionata come migliore presenta un numero di neuroni nel hidden layer pari a 10.

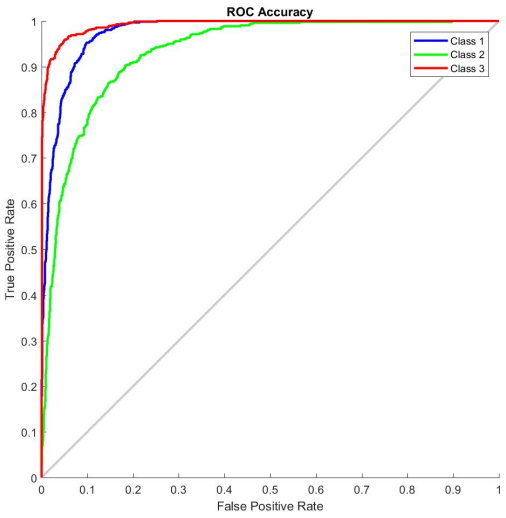




Osservando e valutando i dati riportati nelle figure si nota, come già detto, che la classe numero due è la classe più affetta da errore. L'accuratezza generale della rete è del 85,8%, mentre l'accuratezza più bassa è del 69,9%.

La seconda rete patternet creata, avente due hidden layer, di cardinalità 93 e 4 offre le seguenti prestazioni.

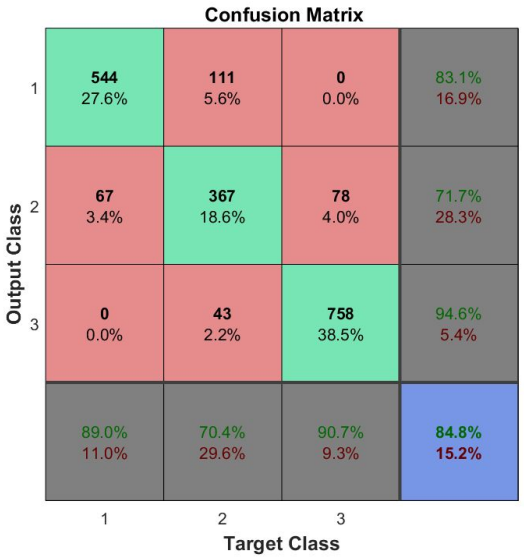


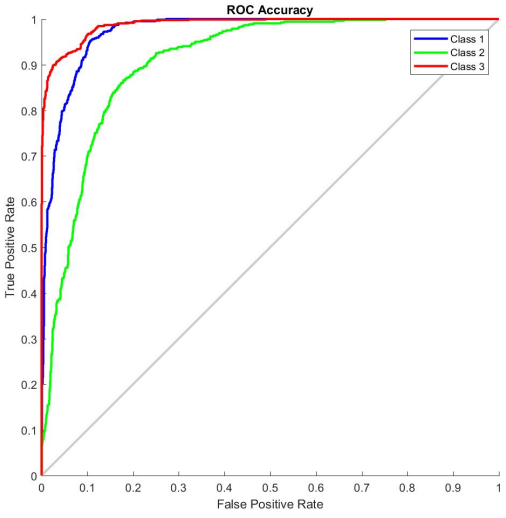


In questo caso, l'introduzione di un ulteriore hidden layer ha incrementato l'accuratezza generale delle rete. In particolar modo, rispetto alla rete con un solo hidden layer, l'accuratezza della classe numero 2 risulta essere significativamente migliorata.

7.2.2 Reti fitnet

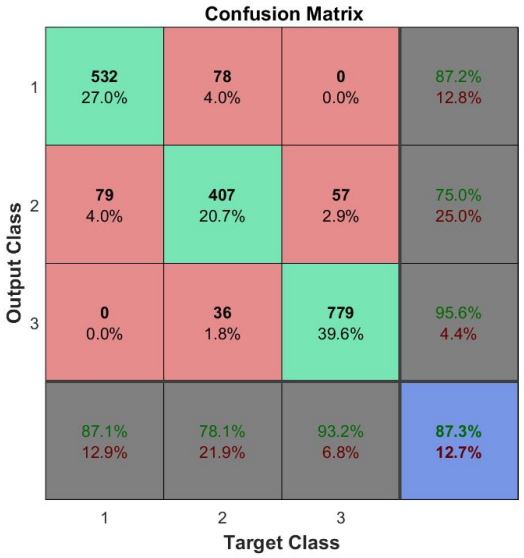
La rete fitnet, con un solo hidden layer, avente le migliori prestazioni presenta un numero di neuroni "hidden" pari a 23.

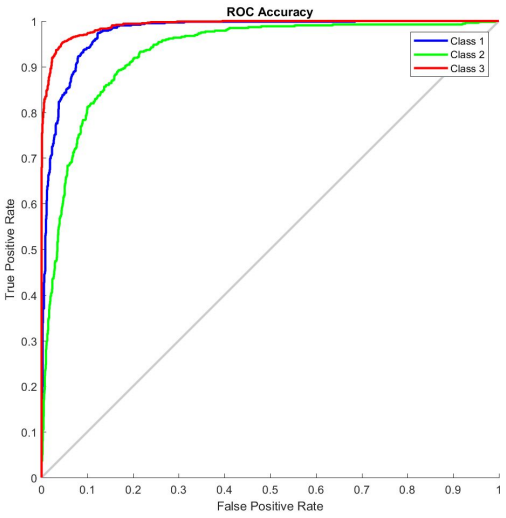




L'accuratezza totale è in questo caso inferiore a quelle delle due reti patternet, infatti in questo abbiamo "solamente" 84,8%.

Per la rete composta da due hidden layer, il numero di neuroni ottimale è pari a 63 per quanto riguarda il primo layer e 9 per quanto riguarda il secondo. Riportiamo sotto, confusion matrix e curve ROC.

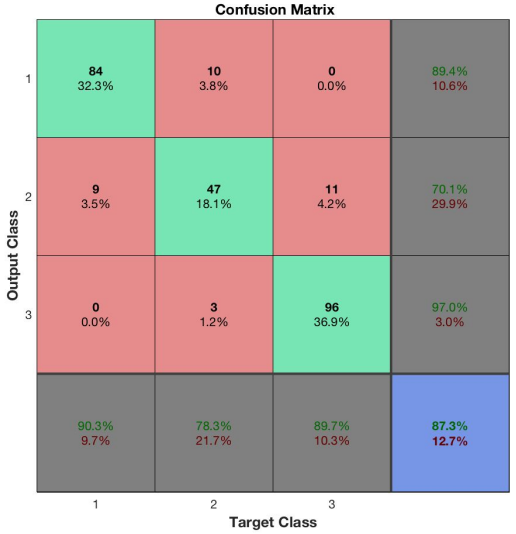


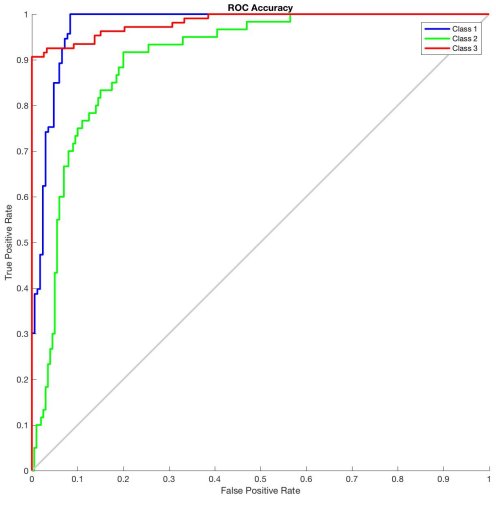


Come si nota, valutando l'accuratezza totale, questa rete offre le migliori prestazioni fino ad ora. E' importante ricordare che le reti considerate fino ad ora non sono state selezione come migliori in funzione dell'accuratezza totale. La loro selezione, come gia spiegato, è stata realizzata tramite l'utilizzo di un sistema fuzzy (6.3) basato su una specifica logica. Questa logica tende a favorire classi con accuratezza della classe 2 maggiore.

7.3 Performance RBF

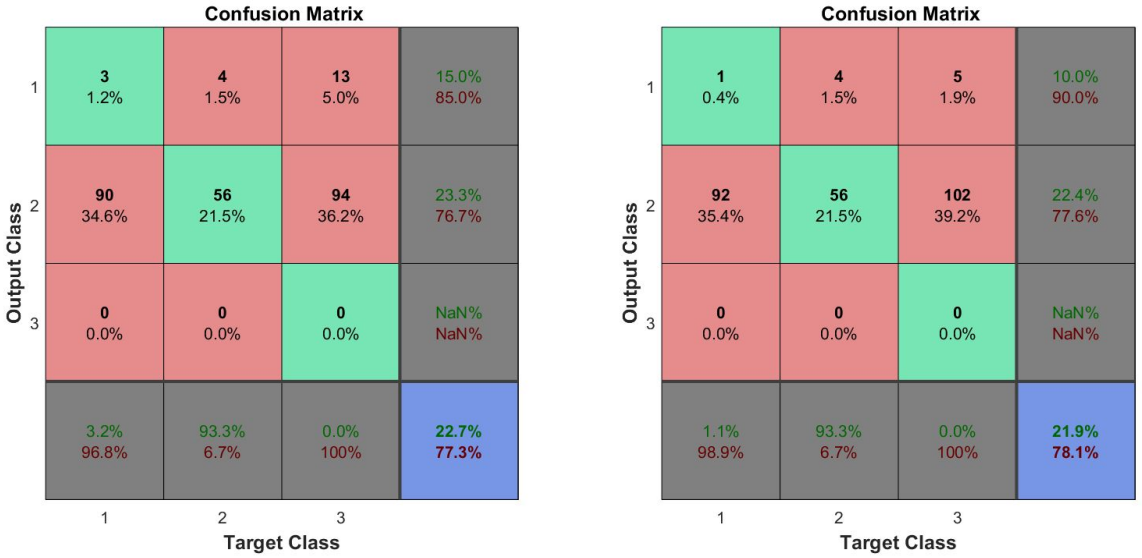
La prima radial basis function network ha un totale di 87 neuroni. Le prestazioni offerte sono riassunte nei seguenti grafici.





Le prestazioni di questa rete sono inferiori alle prestazioni offerte ad ogni tipo di rete MLP.

Inoltre, per quanto riguarda le reti RBF, questa rete risulta essere di gran lunga la migliore. Infatti le prestazioni delle altre due reti, come si evince valutando le confusione matrix riportate qui sotto, non sono assolutamente accettabili.



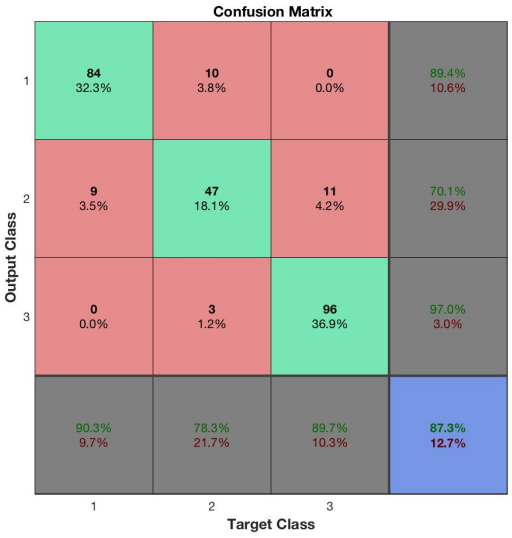
Le prestazioni di queste due reti non sono assolutamente accettabili, quasi tutti gli input vengono associati alla classe di similarità numero 2. Il degrado delle performance è in questo caso causato dalla forte riduzione del dataset iniziale. La funzione *newrbe* infatti, crea un neurone per ogni vettore in input. Sarebbe stato computazionalmente proibitivo passare lo stesso numero di input utilizzato per la prima RBF. Le RBF create utilizzando la *newrbe* non sono assolutamente utilizzabili per la risoluzione di un problema di questa natura.

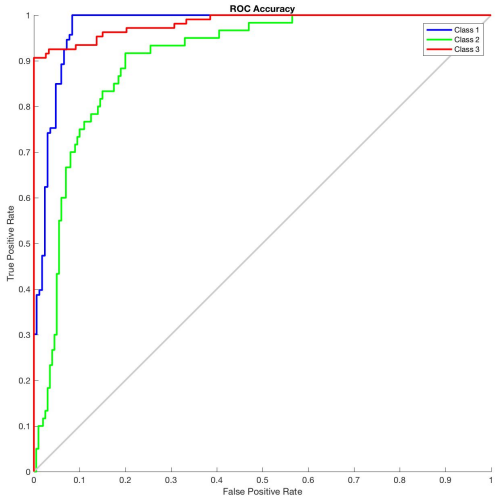
7.4 Conclusioni

Utilizzando il sistema fuzzy trattato in (6.3) abbiamo ottenuto i seguenti risultati.

RETE	FUZZY VALUE	ACCURACY 2
MLP_patternet_1	MEDIUM (0.5)	69,9%
MLP_patternet_2	MEDIUM (0.5)	76,0%
MLP_fitnet_1	MEDIUM (0.5)	70,4%
MLP_fitnet_2	MEDIUM (0.5)	78,1%
RBF_1	**MEDIUM (0.5)**	**78,3%**
RBF_2	MEDIUM-LOW (0.3018)	93,3%
RBF_3	MEDIUM-LOW (0.3018)	93,3%

Le performance delle reti MLP sono risultate in questo contesto generalmente migliori delle performance offerte dalle reti RBF. La miglior rete in assoluto, è la rete di tipo RBF creata tramite la funzione *newrb,* centri inizialmente selezionati in modo causale e spread variabile normalizzato ed avente 87 neuroni. Le performance di questa sono state valutate uguali a quelle di altre reti dal nostro sistema fuzzy. La scelta tra le migliori è stata quindi dettata dal valore sull'accuratezza della classe 2. Riproponiamo di seguito curve ROC e confusion matrix relative a questa rete.





7 Interfaccia grafica

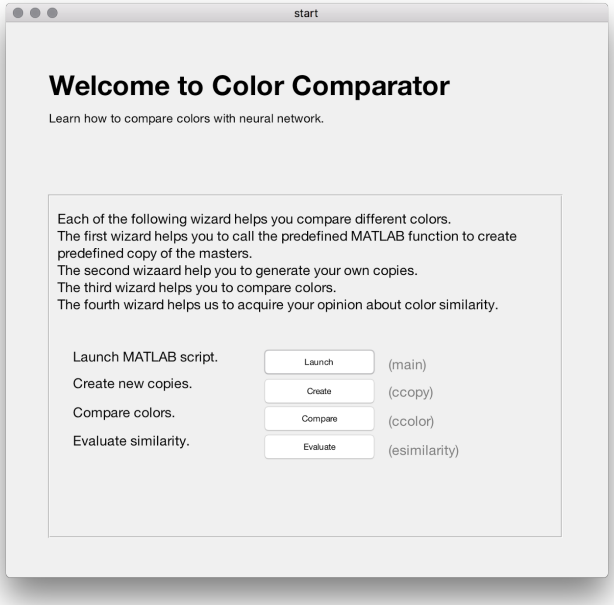
7.1 Introduzione

In questo capitolo andremo a descrivere il tool grafico da noi sviluppato con l'ausilio di GUIDE (ambiente di sviluppo GUI rilasciato da Matlab), esaminando in particolare le funzioni che questo offre all'utente.

7.2 Start interface

Il tool viene aperto con il comando start (da digitare all'interno della cartella di progetto). L'interfaccia che verrà visualizzata all'utente permette di accedere a 4 principali funzionalità:

- Eseguire uno script per la generazione automatica delle copie
- Visualizzare una GUI dedicata alla creazione di copie personalizzate
- Visualizzare una GUI dedicata alla comparazione di copie
- Visualizzare una GUI che permette di valutare la similarità fra un master e le relative copie

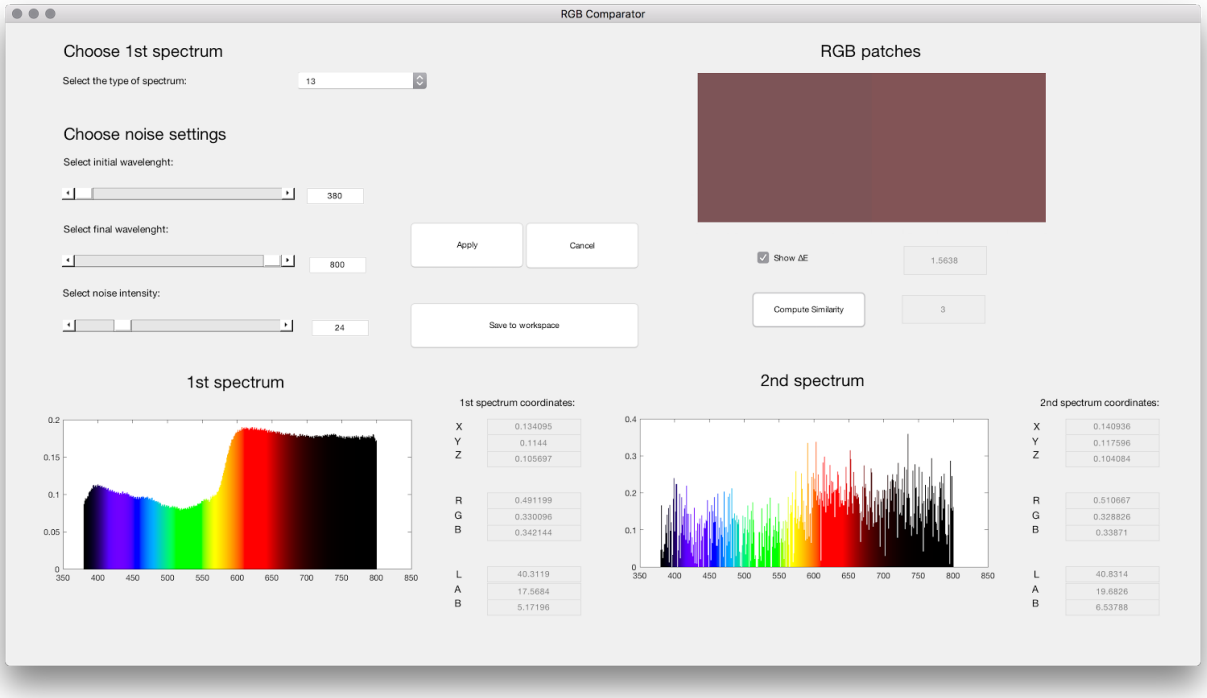


7.3 CCopy interface

Questa interfaccia permette all'utente di creare copie di un determinato master. Una volta scelto il master, l'utente potrà decidere sia l'intervallo di lunghezze d'onda su cui applicare il disturbo sia l'intensità del disturbo stesso. Una volta generata la copia (premendo il tasto “apply”), l'interfaccia visualizza:

- Spettri di master e copia
- Coordinate XYZ, RGB e L*a*b* di master e copia
- Patch con i colori RGB di master e copia affiancati
- Distanza L*a*b* (opzionale)
- Percentuale di similarità prodotta dalla nostra rete neurale

L'interfaccia appena descritta viene riportata di seguito:



7.4 CColor interface

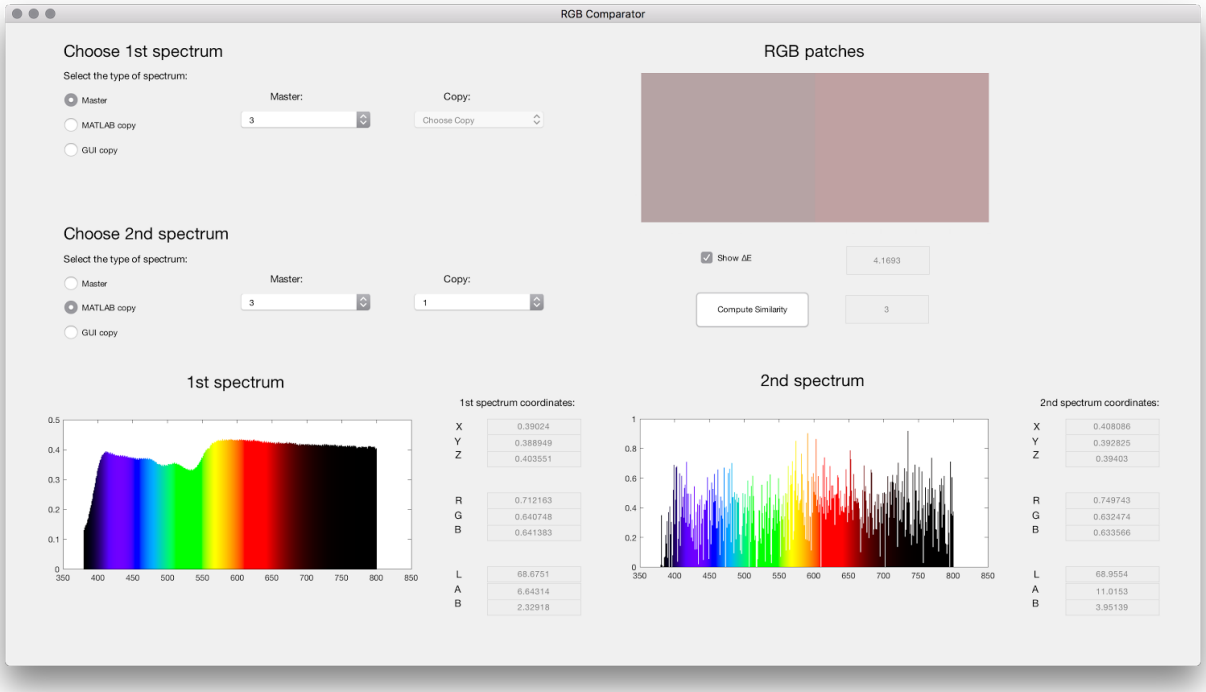
Questa interfaccia permette all'utente di comparare due diverse patch selezionabili tra:

- Master
- Copie generate automaticamente
- Copie create dall'utente tramite CCcopy interface

Per ogni coppia di patch selezionata il tool visualizza:

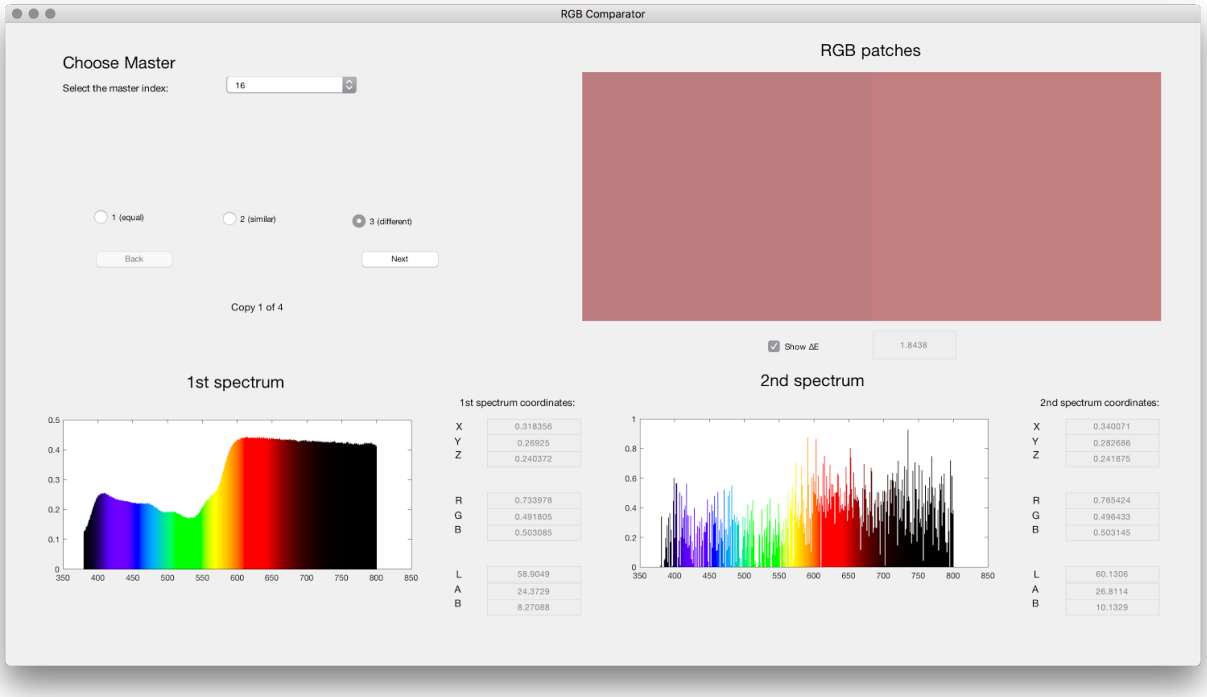
- Spettri
- Coordinate XYZ, RGB e L*a*b*
- Patch con i colori RGB affiancati
- Distanza L*a*b* (opzionale)
- Percentuale di similarità prodotta dalla nostra rete neurale

L'interfaccia appena descritta viene riportata di seguito:



7.5 Esimilarity interface

Questa interfaccia grafica è stata pensata con lo scopo di valutare soggettivamente la similarità tra due patch. Il tool mette a disposizione un menù a tendina con cui è possibile selezionare il master desiderato. Ancora una volta l'interfaccia visualizzerà colori, coordinate e spettri di master e copie. La funzionalità introdotta da questa interfaccia è la possibilità di scorrere tra le diverse copie di un master assegnando di volta in volta il grado similarità percepito. L'interfaccia appena descritta viene riportata di seguito:



Da notare che questa interfaccia è stata utilizzata per la valutazione soggettiva tratta nel paragrafo 3.3.