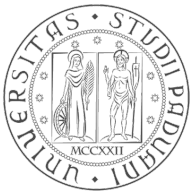


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Support Vector Machines per la classificazione e loro applicazione al controllo di qualità di prodotti siderurgici

Laureando

Luca Moscatelli

Relatore

Prof. Alessandro Chiuso

Co-relatore

Ing. Daniele Fornasier

Anno Accademico 2014/2015

Indice

Capitolo 1

Introduzione

Questa tesi nasce dall'incontro tra l'azienda friulana *Video Systems*, che si occupa di sviluppo di nuove tecnologie di visione artificiale e di sistemi di visione e l'Università di Padova. Video Systems progetta e realizza sensori, sistemi di visione e di controllo qualità ad alto contenuto tecnologico per tutti i principali settori industriali, tra cui: vetro, siderurgica, automazione, robotica, automotive. Implementa inoltre soluzioni per il controllo di processo con sistemi di visione ad alte prestazioni. É in quest'ultimo ambito che nasce tale collaborazione. Video Systems aveva due esigenze:

- l'apprendimento e lo studio delle capacità e delle potenzialità del metodo di machine learning Support Vector Machines (SVM);

- la ricerca di soluzioni efficienti per due problemi di controllo di qualità che l'azienda si trovava ad affrontare e per i quali non avevano ancora trovato soluzioni sufficientemente performanti.

1.1 Presentazione dei problemi

I problemi che si sono affrontanti rientrano in problemi di controllo qualità ossia, si ha la necessita, data l'immagine dell'oggetto, di determinare e segnalare la presenza di difetti o di residui di lavorazione. La tipologia dell'oggetto e dei residui varierà in base al problema affrontato.
In entrambi i casi l'immagine viene acquisita dal sistema di visione all'interno di un cilindro metallico. Nel primo viene catturato il fondo di tale oggetto mentre nel secondo un intramezzo circolare incollato tramite resina. Entrambi gli oggetti esaminati avranno forma circolare.
I due problemi differiscono per diversi aspetti:

- *la tipologia dell'oggetto esaminato*: nel primo problema si esamina il fondo del cilindro che può risultare di due tipologie distinte con caratteristiche diverse; nel secondo caso si esamina un'unica tipologia di oggetto che possiede una specifica forma.

- *ciò che si vuole determinare*: nel primo caso si vogliono individuare solo alcune anomalie rispetto all'oggetto di base andando a scartare altre anomalie non considerate difetto; nel secondo, qualsiasi oggetto o sostanza che non risulti appartenere alla struttura base dell'oggetto esaminato.

A livello teorico possiamo ricondurre i sistemi di controllo di qualità implementati ad uno schema generale, la cui struttura è riportata in Figura 1.1.

Si possono individuare quattro sotto problemi:

Fig. 1.1: Schema generale del sistema di visione per il controllo qualità.

Image Processing Obiettivo di questa fase è ripulire l'immagine da eventuale rumore e restituire un'immagine normalizzata in scala di grigio, in cui viene selezionata la parte dell'immagine utile a raggiungere il nostro scopo. In entrambi i casi viene individuata la sezione circolare dell'immagine che comprende l'oggetto da esaminare. La parte esterna alla sezione circolare viene scartata.

Individuazione regioni di interesse (candidati) I due obiettivi prefissi per entrambi i problemi sono: determinare se nell'immagine sono presenti anomalie e segnalare la loro posizione nell'immagine. Al fine di raggiungere il secondo obiettivo l'analisi svolta non riguarderà l'intera immagine, ma si andranno ad individuare aree dell'immagine di maggior interesse che verranno chiamate *candidati*. Nel primo problema si adotterà un algoritmo di *blob detection*, mentre nel secondo si utilizzerà la tecnica delle *sliding windows*.

Estrazione feature vector Le features sono un insieme di misure o di proprietà qualitative che permette di caratterizzare completamente l'oggetto, nel nostro caso il candidato, ai fini del riconoscimento. Risultano insensibili a variazioni non significative presenti sulle istanze dell'oggetto e consentono di discriminare tra istanze di oggetti diversi. In letteratura le features vengono divise in:

- *Global features* quando le features vengono calcolate rispetto all'intera immagine o per la medesima area dell'immagine;
- *Local features* quando le features vengono calcolate per una specifica area dell'immagine determinata, area che è stata individuata come area notevole dell'immagine;
- *Pixel-levl features*: quanto le features vengono determinate per ciascun pixel dell'immagine;

Nei progetti useremo solamente local features che daranno informazione sulla forma dei nostri candidati, sul colore o sul gradiente. Utilizzeremo tali descrittori per formare il vettore, **feature vector**, che utilizzeremo come ingresso al classificatore.

Classificatore feature vector I feature vectors saranno utilizzati come ingresso al classificatore implementato tramite Support Vector Machine (SVM), al fine di determinare se il candidato appartenga alla classe dei difetti o non difetti.

In accordo con Video Systems si è scelto di testare le potenzialità delle Support Vector Machines (SVM). Tale tecnica fa parte di un insieme più ampio di algoritmi chiamati *machine learning*. L'apprendimento automatico rappresenta una delle aree fondamentali dell'intelligenza artificiale e si occupa della realizzazione di sistemi e algoritmi che si basano su osservazioni come dati per la sintesi di nuova conoscenza. L'apprendimento avviene catturando caratteristiche di interesse (features) provenienti da esempi, strutture dati o immagini per valutare le relazioni tra le variabili osservate. Le macchine a vettori di supporto sono state sviluppate negli anni '90 da Vladimir Vapnik. Tale tecnica risulta molto potente e per tale motivo negli ultimi anni è stata utilizzata per risolvere un varietà molto ampia di applicazioni. La scelta è ricaduta su questa tecnica per diversi motivi:

- lo sviluppo che questa tecnica ha avuto con ottimi risultati negli ultimi anni nell'ambito della visione computazionale
- la numerosità dei problemi che consente di affrontare quali problemi di classificazione di classi linearmente separabili o di classi non linearmente separabili e problemi di regressione;

Nei paragrafi successivi descriveremo in maniera dettagliata i due problemi affrontati, approfondendo la fase di image processing ed introdurremo le fasi successive che saranno trattate più nel dettaglio nei prossimi capitoli.

1.2 Primo problema

Il primo problema che si vuole risolvere è l'individuazione e la segnalazione di difetti e detriti di lavorazioni sul fondo di cilindri metallici. I fondi di tali oggetti risultano di due tipologie distinte che si differenziano per la forma: a fondo piatto che sarà indicata con la sigla STD, ed a fondo ricalcato che sarà indicata con la sigla REV.
Innanzitutto è necessario specificare alcune caratteristiche delle immagini e degli oggetti analizzati:

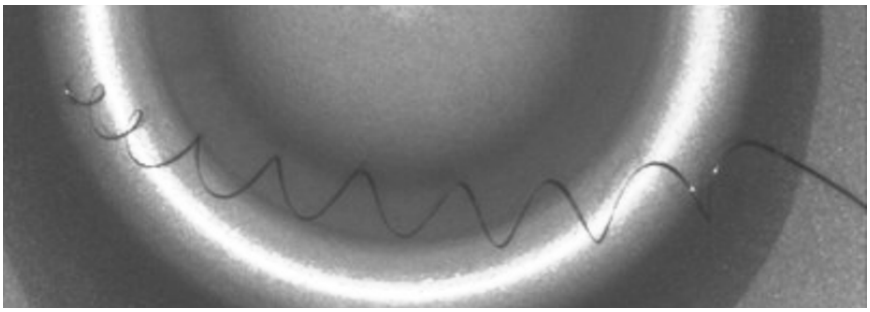
- non è a disposizione un modello base dell'immagine di tali oggetti. Le immagini non vengono scattate esattamente alla medesima altezza e quindi si ottengono immagini più o meno zoomate.
- a volte l'immagine risulta sfocata a causa della presenza di vapore acqueo al momento della sua acquisizione. Tale problema si riscontra esclusivamente per la prima tipologia (STD) ed è una conseguenza del processo di fabbricazione. La presenza del vapore acqueo, assimilabile ad un problema di sfocatura, è un problema di notevole importanza poiché come vedremo ha conseguenze in entrambe le fasi del problema: sia per quanto riguarda l'individuazione dei candidati, in quanto non si ha una buona individuazione dei bordi, sia nella fase di classificazioni, poiché i colori ne risultano falsati.

Per la comprensione e per le successive fasi del problema è necessario definire il concetto di *difetto*.
Si considera difetto:

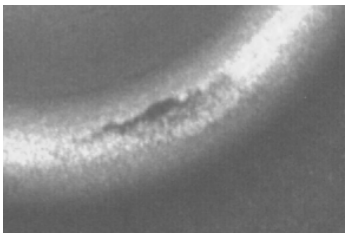
- qualsiasi agente esterno: trucioli, polvere ferrosa o altri residui di lavorazione, Figura 1.2;
- la presenza di grossi quantitativi di liquidi, Figura 1.3;
- i difetti intrinseci del prodotto, come ammaccature o sporgenze, Figura 1.4.

Come si può notare dalle immagini sopra riportate, esiste una varietà ampia di difetti che non può essere accomunabile né per forma né per dimensione. Inoltre non c'è un difetto base a cui ricondursi. Per esempio, se si prendono in considerazione le due immagini che mostrano le tipologie di trucioli, si può notare la loro diversità di dimensione. Il primo truciolo presenta delle arricciature, al contrario del secondo caso che non le presenta. Inoltre la lunghezza dei due trucioli è notevolmente diversa. Se si confrontano le immagini delle deformazione sul bordo, si può notare che una è in rilievo mentre la seconda è una ammaccatura. Il liquido, per sua definizione, non ha forma e quindi se ne possono avere infinite casistiche.

In questo primo problema i primi due step dello schema di Figura 1.1, ossia image processing e l'individuazione dei candidati, vengono svolti dal software fornito da Video Systems. Come riportato in Figura 1.5, l'immagine a colori viene convertita in scala di grigio e viene individuata la regione circolare contenente l'oggetto. Il resto dell'immagine viene scartato e nelle fasi successive non verrà più presa in considerazione. Alla sezione circolare individuata viene applicata una



(a) Truciolo

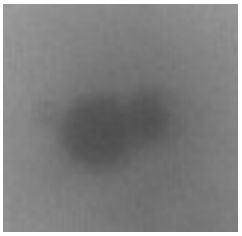


(b) Polvere ferrosa

(c) Truciolo

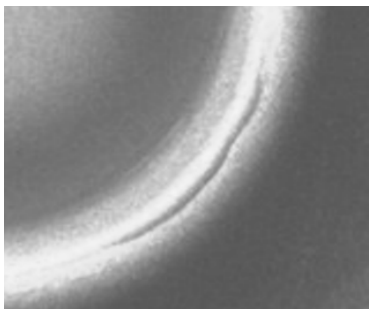
Fig. 1.2: Primo problema: prima tipologia di difetti.

(a) Liquido

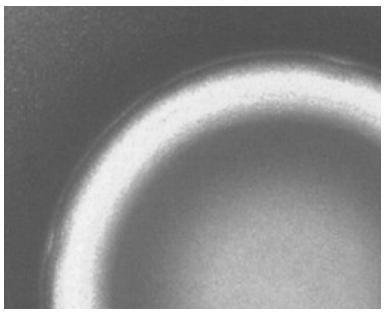


(b) Liquido

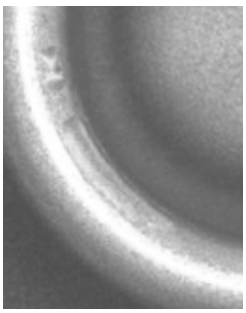
Fig. 1.3: Primo problema: seconda tipologia di difetti.



(a) Deformazione sul bordo del fondo, sporgenza



(b) Deformazione sul bordo del fondo, difetti ad S



(c) Deformazione sul bordo del fondo, ammaccatura

Fig. 1.4: Primo problema: terza tipologia di difetti.

trasformazione polare in modo da ricavare l'immagine dell'apertura polare dell'oggetto esaminato. I colori dell'immagine vengono normalizzati per riga e colonna prima di applicare l'inverso della trasformazione precedente in modo da ricondursi alla forma circolare. Come ultimo step viene eseguito l'algoritmo di blob detection.
Blob detection o riconoscimento di regioni è una tecnica che ha come obiettivo di rilevare punti e/o regioni in una immagine che differiscono in proprietà come luminosità o colore comparata

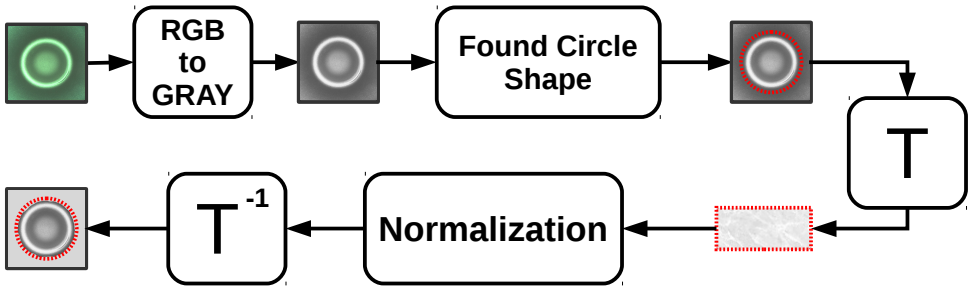


Fig. 1.5: Primo problema, schema fase di image processing.

con l'ambiente. In questa fase all'immagine normalizzata ricavata al punto precedente, viene applicato un algoritmo di tresholding con doppia soglia. Tale algoritmo individua le parti dell'immagine che saranno prese in considerazione come possibili candidati di difetti, restituendo un'immagine binarizzata. Tale blocco lo possiamo descrivere come una funzione o un filtro che agisce in più passi: in un primo luogo trasforma l'immagine da scala di grigi, datagli in ingresso, in un'immagine binarizzata (step 1 di Fig. 1.6). Successivamente individua tutte le regioni connesse dell'immagine binarizzata (step 2 diFig. 1.6) andando a scartare le regioni connesse di dimensione inferiore a 40 pixel. Infine come candidati si considerano le regioni dell'immagine passata in ingresso contenuti nel più piccolo rettangolo che inscrive le regioni precedentemente determinate, tali rettangoli sono chiamati bounding box (step 3 di Fig. 1.6).

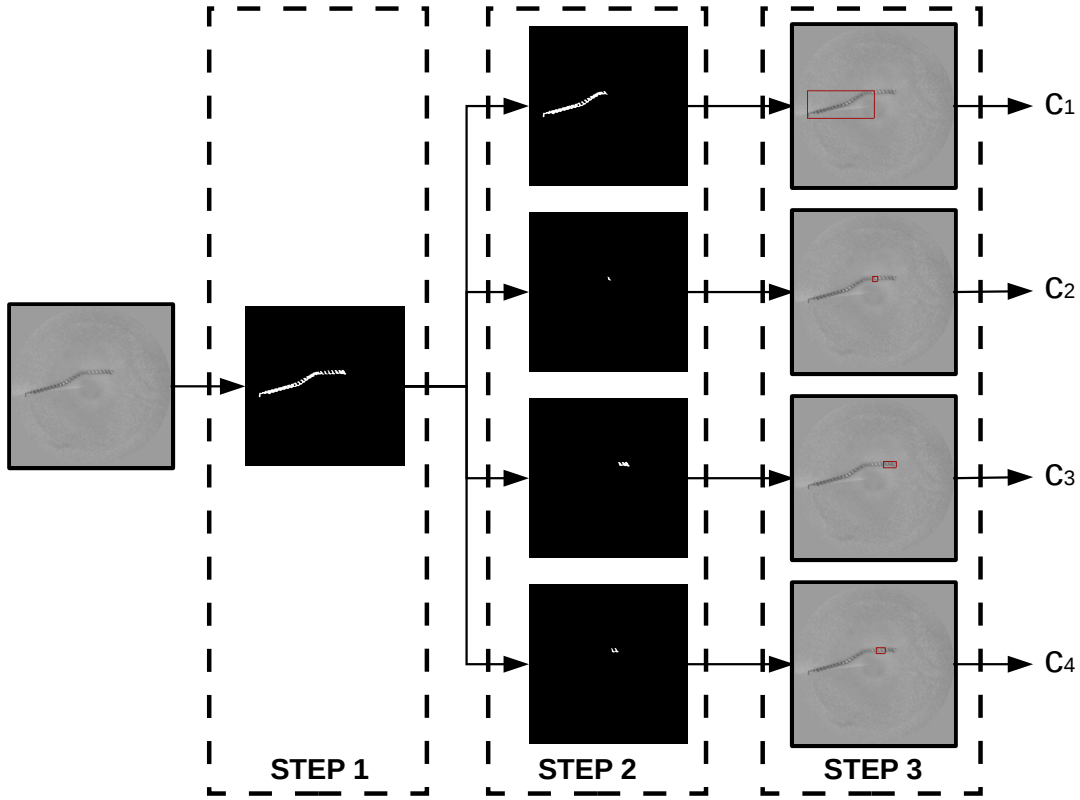


Fig. 1.6: Primo problema, Blob detection

I candidati così ottenuti sono il punto di partenza del nostro primo problema. Da essi, verranno estratti descrittori di colore e di forma che andranno a formare i *feature vectors*, vettori

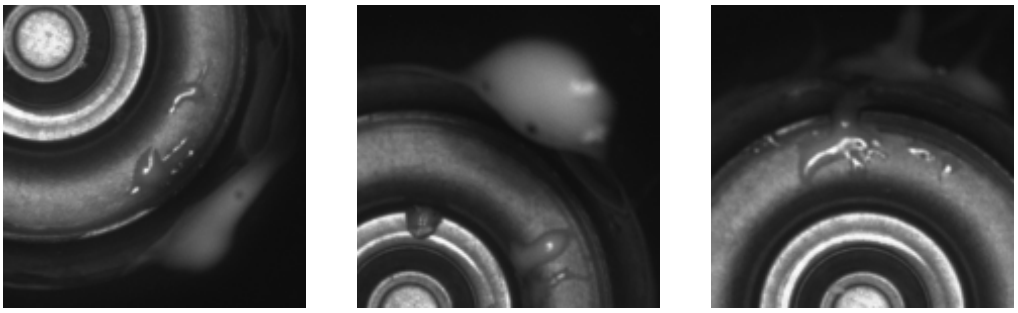


Fig. 1.7: Esempi di difetti del secondo problema

d'ingresso dei classificatori implementati attraverso le SVM.

1.3 Secondo problema

Il secondo problema, come il primo, vuole determinare la presenza di scarti di lavorazione. In questo caso con scarti si intende la presenza di rimanenze di resina utilizzata per bloccare l'oggetto all'interno del cilindro dal quale viene catturata l'immagine da analizzare.
Questo secondo problema presenta aspetti più semplici rispetto al primo. Innanzitutto si esamina solo una tipologia di oggetto. In secondo luogo si ha una tipologia base dell'oggetto ed infine si considererà difetto tutto quello che non appartiene al modello base.
L'oggetto ha forma circolare e presenta una serie di circonferenze inscritte. Ciò che si vuole è determinare la presenza o meno di resina sopra a tale oggetto. Alcuni esempi di difetti sono riportati nelle immagini di Figura 1.7.
Anche in questo problema l'immagine viene inizialmente elaborata al fine di determinare la sezione circolare dell'oggetto da controllare. L'immagine di partenza è un immagine rgb di dimensioni 1944×2552. Le dimensioni originale portano ad ottenere un'informazione ridondante, per tale motivo l'immagine è scalata di un fattore 0.25 e convertita in scala di grigi. Da quest'ultima, sfruttando il contrasto di colore tra il background dell'oggetto, di colore scuro, e l'oggetto che presenta un intensità maggiore di colore, viene estratto il blob delle dimensioni dell'oggetto. Da tale immagine si ricava la circonferenza che racchiude l'oggetto da esaminare ottenendo così la sezione circolare dell'immagine desiderata. Si utilizza quindi una trasformazione polare ottenendo l'immagine dell'apertura polare dell'oggetto. Tale immagine presenta caratteristiche di gradiente molto particolari che andremo a sfruttare tramite l'Histogram of Oriented Gradient (HOG). Le HOG features formeranno i vettori di ingresso al classificatore.

1.4 Struttura tesi

Nel prossimo capitolo si studierà la teoria alla base delle Support Vector Machine. Successivamente nei capitoli tre e quattro, verranno affrontati più nel dettaglio i due problemi e verranno presentati i risultati dei test ottenuti. Infine nel capitolo 5 si trarranno le conclusioni del nostro elaborato.

Capitolo 2

Supervised learning

Quando si deve risolvere un problema pratico molte volte occorre determinare la relazione esistente tra un certo output e alcuni valori di ingresso che possono essere descritti esplicitamente. Cercare di determinare e di implementare queste relazioni può risultare molto difficile e, a volte, quasi impossibile, soprattutto se si ha un numero di output e di input molto grande. Una strategia alternativa per risolvere queste tipologie di problemi è quella di far apprendere al computer tale relazione tramite degli esempi che utilizzano la coppia di input e output o solamente input. Gli approcci che usano esempi di informazione per allenare l'algoritmo sono chiamati *machine learning*. Questa tipologia di algoritmi è utilizzata su problemi di Finanza, analisi di mercato, analisi e previsioni di fluttuazione azionarie, nel campo della medicina, analisi di motivi di DNA/RNA, identificazione di nuovi geni o nel campo della visione artificiale, classificazioni di immagini e in molti altri campi.
Tali metodi si dividono in base al tipo d'informazione che usano per allenare gli algoritmi:

- le tecniche che usano coppie di input/output per allenare l'algoritmo sono chiamati *supervised learning* e tali coppie prendono il nome di *training data*;
- le tecniche che usano solo gli input per allenare l'algoritmo vengono chiamate *unsepervised learning*

Nel nostro studio ci concentreremo sulla prima tipologia. I training example tipicamente sono l'effetto di una funzione che mappa gli input negli output. Quando questa funzione esiste viene chiamata *target function*. La stima di tale funzione è la soluzione del problema di apprendimento. Nel caso di problemi di classificazione tale funzione è chiamata *decision function* ed è scelta attraverso un insieme di funzioni che mappa lo spazio di input nel codominio. Tale insieme di funzioni è chiamato *hypotheses*. Gli algoritmi supervised learning presentano quindi due ingredienti fondamentali:

- l'insieme di funzioni tra cui verrà stimata la target function (hypothesis space)
- l'algoritmo che usa i training data come ingresso per selezionare una delle hypotheses function.

Cerchiamo ora di definire in modo più rigoroso i concetti finora esposti.
Come detto in precedenza le tecniche di supervised learning hanno bisogno di un set di dati di training S costituito da m osservazioni di coppie di dati $(x^{(i)}, y^{(i)})$, dette training example, dove $x^{(i)}$ denota l'input o features ($x^{(i)} \in X$ dove X è lo spazio delle features) e $y^{(i)}$ rappresenta l'output o labels ($y^{(i)} \in Y$ è lo spazio dell'output).

$$S = \left\{ \left(x^{(1)}, y^{(1)} \right), \ldots, \left(x^{(m)}, y^{(m)} \right) \right\} \quad x^{(i)} \in X \quad y^{(i)} \in Y \tag{2.1}$$

L'obiettivo delle tecniche di apprendimento supervisionato è, dato un training set S, allenare una funzione $h : X \to Y$, hypothesis function, affinché sia un buon predittore del valore di y

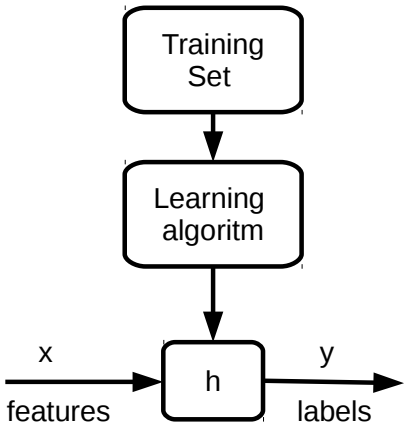


Fig. 2.1: Supervised Learning

Quando la variabile di output y che vogliamo predire è continua si parla di problema di regressione, quando la variabile di output assume un finito numero discreto di valori discreti allora si parla di problema di classificazione. Le Support Vector Machines sono algoritmi di apprendimento che usano come hypothesis space, spazi di funzioni lineari con dimensione delle features molto grandi e che selezionano la target function andando a massimizzare il margine tra le classi di classificazione. Le SVM quindi possono essere impiegate per risolvere problemi di classificazione binaria o di multi-classificazione, sebbene possano essere impiegate anche nella regressione. In questo capitolo vengono illustrati i problemi di classificazione e viene effettuata una trattazione matematica sulle SVM.

2.1 Problema di classificazione e SVM

La classificazione consiste nel problema di identificare a quale categoria appartenga un nuovo dato osservato, sulla base di un set di dati di training, di cui è nota la categoria di appartenenza. Consideriamo ora il caso più semplice di classificazione, ossia la classificazione binaria in cui le variabili di label possono assumere solo due valori 1, positive class, o -1, negative class.

$$S = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\} \quad x^{(i)} \in \mathbb{R}^2 \quad y^{(i)} \in \{-1, 1\} \tag{2.2}$$

Se rappresentiamo i dati su un piano si possono verificare due situazioni. Nel caso in cui sia possibile dividere le due classi di dati con una retta o un iperpiano, Fig. 2.2, si tratta di dati linearmente separabili. Nel caso contrario si parlerà di dati non linearmente separabili, Fig. 2.3.

Rimanendo nel caso di una classificazione bilineare con dati separabili, proviamo a spiegare a parole cosa si intende per confidenza di una predizione. Consideriamo la Figura 2.4, nella quale i cerchi rappresentano i punti appartenenti alla positive class e i quadrati quello appartenenti alla negative class, e sia $\theta^T x = 0$ l'equazione dell'iperpiano che divide i due insiemi. Prendiamo ora in considerazione i tre punti A, B e C. Il punto A è molto distante dall'iperpiano che separa le due classi, quindi possiamo dire che tale punto appartiene alla positive class con un buon grado di confidenza. Al contrario, il punto C è molto vicino a tale linea e non possiamo essere più così sicuri che tale punto posso appartenere davvero alla positive class. Si può sostenere quindi che il grado di confidenza di questa predizione è piuttosto basso. Il punto B è un caso intermedio tra il punto A e il punto C. Intuitivamente quindi, più un punto è distante dall'iperpiano di separazione, maggiore sarà il grado di confidenza della nostra predizione.Formalizziamo ora tale concetto.

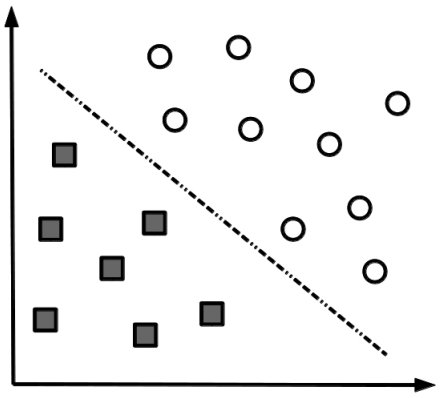


Fig. 2.2: Insieme linearmente separabile

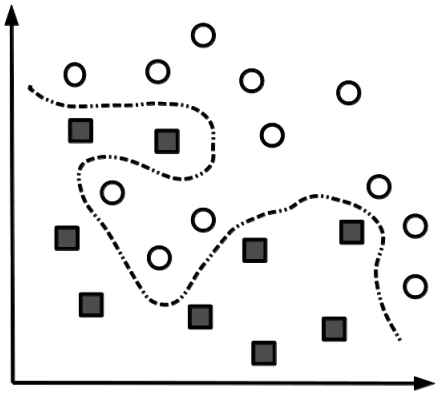


Fig. 2.3: Insieme non separabili

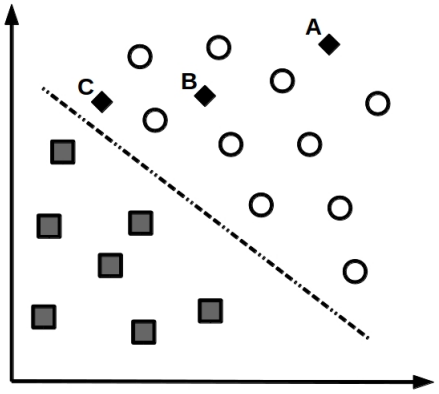


Fig. 2.4: Classificazione binaria

A tale scopo definiamo la nostra hypothesis function :

$$h_{w,b}(x) = g(w^T x + b) = \begin{cases} 1 & \text{se} \quad w^T x + b \geq 0 \\ -1 & \text{se} \quad w^T x + b < 0 \end{cases} \tag{2.3}$$

Questo tipo di hypothesis function divide lo spazio degli input, X, in due parti e l'iperpiano di separazione è dato dall'equazione $w^T x + b = 0$. Il vettore w rappresenta quindi il vettore perpendicolare all'iperpiano, mentre se si varia b si sposterà l'iperpiano all'interno dell'input space mantenendolo parallelo a se stesso. Dato un training example $(x^{(i)}, y^{(i)})$ definiamo **functional margin** di (w, b) rispetto al training example :

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

Nel caso in cui $y^{(i)} = 1$, cioè quando $x^{(i)}$ appartiene alla positive class, il nostro desiderio è che $w^T x^{(i)} + b$ abbia un valore positivo elevato, al fine di ottenere una predizione corretta con un buon grado di confidenza. Al contrario, nel caso in $y^{(i)} = -1$ per ottenere una classificazione corretta dobbiamo ottenere $w^T x + b < 0$. Quindi, generalizzando, se $y^{(i)}(w^T x + b) > 0$ la catalogazione risulta corretta e maggiore è il valore del functional margin, maggiore è la confidenza della nostra classificazione. Dato un training set S definiamo **function margin** di (w, b) rispetto ad S il più piccolo functional margin dei training example:

$$\hat{\gamma} = \min_i \hat{\gamma}^{(i)}$$

.

Consideriamo la Figura 2.5 che mostra l'iperpiano che divide le due classi e il vettore w. Tale vettore è ortogonale all'iperpiano. Infatti se consideriamo due punti dell'iperpiano x' e x'', con $x' \neq x''$, essi per definizione devono verificare l'equazione $w^T x' + b = 0$ e $w^T x'' + b = 0$. Se ora sottraiamo le due equazioni otteniamo come risultato $w^T(x' - x'') = 0$ dove il vettore $x' - x''$ che unisce i due punti è diverso da 0 per ipotesi iniziale ed appartiene all'iperpiano. Ne consegue, quindi, che il vettore w deve essere necessariamente ortogonale all'iperpiano.
Sia ora il punto A, la proiezione nel piano dell'input $x^{(i)}$, appartenente alla positive class e sia $\gamma^{(i)}$, la distanza di $x^{(i)}$ con l'iperpiano (il segmento AB), vogliamo determinare l'equazione di tale distanza. Consideriamo il versore $\frac{w}{\|w\|}$. Possiamo scrivere $B = x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|}$. Come punto appartenente all'iperpiano deve soddisfare l'equazione:

$$w^T B + b = w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|} \right) + b = 0$$

Risolvendo per $\gamma^{(i)}$ si ottiene

$$\gamma^{(i)} = \left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|}$$

Tale definizione è valida per il caso di un training example appartenente alla positive class. Generalizzando il risultato ottenuto possiamo definire che il margine geometrico di (w, b) rispetto al training example $(x^{(i)}, y^{(i)})$ risulta:

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right)$$

Infine definiamo **geometric margin** di (w, b) rispetto al training set S, eq. (2.2), il più piccolo margine geometrico individuato tra i training example:

$$\gamma = \min_i \gamma^{(i)}$$

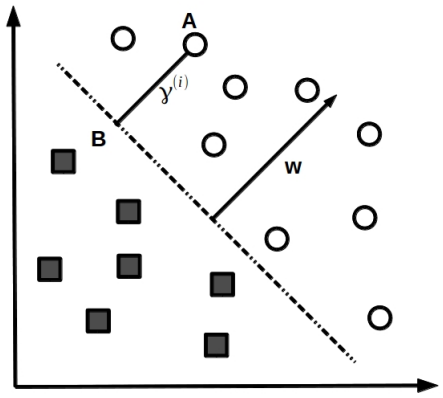


Fig. 2.5: Classificazione binaria

Prima di proseguire osserviamo alcuni aspetti:

- se $\|w\| = 1$ il geometric margin e il functional margin coincidono.

- se scaliamo di uno stesso fattore i parametri w e b, ad esempio consideriamo $2w$ e $2b$, il geometric margin non varia. Il geometric margin risulta invariante al riscalamento dei parametri.

- se consideriamo l'hyphotesis function (2.3), scalando di uno stesso fattore, $k > 0$, i parametri w e b, il risultato finale non cambia. Infatti il risultato dipende dal segno di $w^T x + b$ e non dal suo valore.

Dalla discussione informale fatta in precedenza, si è concluso che maggiore è il geometric margin, maggiore risulta la confidenza della classificazione. Quindi l'obiettivo è ora quello di determinare l'iperpiano che presenta il più grande geometric margin. Dobbiamo risolvere il seguente problema di massimizzazione:

$$\begin{array}{ll} \max\limits_{\gamma,w,b} & \gamma \\ s.t. & y^{(i)}(w^T x^{(i)} + b) \geq \gamma \\ & ||w|| = 1 \end{array} \tag{2.4}$$

Il vincolo $\|w\| = 1$ garantisce che il geometric margin coincida con il functional margin e ci garantisce che tutti i geometric margin siano maggiori di γ.

Tale problema non ha la forma di un problema di programmazione matematica in quanto né la funzione costo né i vincoli sono funzioni convesse.

Proviamo allora a riscrivere lo stesso problema cercando di ottenere un problema di ottimizzazione convessa, sfruttando la relazione fra functional margin e geometric margin $\left(\gamma = \frac{\hat{\gamma}}{||w||}\right)$ e tenendo bene in mente che il nostro obiettivo è quello di determinare i parametri b e w dell'iperpiano ottimo, ossia l'iperpiano che presenta il più grande geometric margin. Riscriviamo il problema :

$$\begin{array}{ll} \max\limits_{\hat{\gamma},w,b} & \frac{\hat{\gamma}}{||w||} \\ s.t. & y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma} \end{array}$$

dove però la funzione costo risulta ancora una funzione non convessa.

Per ottenere un problema di ottimizzazione convessa sfruttiamo la possibilità di riscalare i parametri w e b al fine di ottenere $\hat{\gamma} = 1$, sapendo che tale riscalatura non ha effetto sul margine geometrico. Infine, sapendo che la minimizzazione di $||w||$ coincide con quella di $||w||^2$, si ottiene il problema :

$$\min_{w,b} \quad \frac{1}{2}||w||^2 \tag{2.5}$$

$$s.t. \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 \tag{2.6}$$

Siamo riusciti a trasformare il problema (2.4) in un problema di minimo con funzione costo quadratica convessa e dei vincoli lineari, la cui soluzione è l'**iperpiano ottimo**.

Di seguito calcoliamo la versione duale del nostro problema di minimizzazione per due ragioni:

- perché i vincoli del problema duale saranno più semplici di quelli del problema primale;

- nella formulazione duale, le features compariranno attraverso prodotti scalari tra le stesse features, ciò consentirà di generalizzare la tecnica al caso di insiemi non linearmente separabili.

Problema Duale

Per determinare la forma duale del problema di minimo 2.6 utilizziamo il metodo dei moltiplicatori di Lagrange. Il problema duale assumerà la forma :

$$\max_{\alpha:\alpha_i\geq 0} \quad \theta_D(\alpha) = \max_{\alpha:\alpha_i\geq 0} \left(\min_{w,b} \ \mathcal{L}(w,b,\alpha) \right) \tag{2.7}$$

dove $\mathcal{L}(w,b,\alpha)$ è la funzione Lagrangiana, che nel nostro caso corrisponde:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$

dove $\alpha_i \geq 0, i = 1, \ldots, m$, sono le variabili duali e w e b sono le variabili primali.

Per determinare il problema duale, dobbiamo per prima cosa calcolare la dual function,cioè il valore di minimo della Lagrangiana al variare delle variabili primali per un valore costante di α:

$$\theta_D(\alpha) = \min_{w,b} \ \mathcal{L}(w,b,\alpha)$$

Dal teorema di Karush-Kuhn-Tucker (Teorema (3) in Appendice) sappiamo che se $(w^\star, b^\star, \alpha^\star)$ sono punti ammissibili, affinché siano punti di minimo globale è necessario che soddisfino le condizioni di Karush-Kuhn-Tucker (KKT):

- (KKT1.w) $\nabla_w \mathcal{L}(w^\star, b^\star, \alpha^\star) = 0 \Longrightarrow w^\star = \sum_{i=1}^{m} \alpha_i^\star y^{(i)} x^{(i)}$
- (KKT1.b) $\nabla_b \mathcal{L}(w^\star, b^\star, \alpha^\star) = 0 \Longrightarrow \sum_{i=1}^{m} \alpha_i^\star y^{(i)} = 0$
- (KKT2) $\alpha_i^\star \left[-y^{(i)}(w^{\star T} x^{(i)} + b^\star) + 1 \right] = 0 \ \ \forall i = 1, \ldots, m$

Allora, sostituendo il vincolo KKT1.w e KKT1.b nell'equazione della Lagrangiana, si ottiene :

$$\theta_D(\alpha) = \sum_{i=1}^{m} \alpha_i - \sum_{j=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)}$$

Siamo riusciti, così, a ricondurci al problema duale di forma:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)} \\ s.t. \quad & \alpha_i \geq 0 \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned} \tag{2.8}$$

Per risolvere il problema iniziale e quindi per determinare w e b coefficienti del iperpiano ottimo, si utilizzerà il problema duale, in quanto problema di più facile soluzione, dal quale si otterrà la variabile $\alpha^\star$ che ottimizza il nostro problema. Da tale variabile andando a sfruttare la condizione KKT1.w siamo in grado di calcolare il valore ottimo della variabile primaria $w^\star$.

Dalla condizione KKT2, ricaviamo :

$$\begin{cases} \alpha_i^\star > 0 & \Longrightarrow y^{(i)}(w^{\star T} x^{(i)} + b^\star) = 1 \\ \alpha_i^\star = 0 & \Longrightarrow y^{(i)}(w^{\star T} x^{(i)} + b^\star) \geq 1 \end{cases}$$

Dall'equazione (KKT1.w) si ricava che i coefficienti di $w^\star$ dipendono solamente dalle feature $x^{(i)}$ per le quali il corrispondente moltiplicatore di Lagrange risulta $\alpha_i > 0$. Nel caso di insiemi

linearmente separabili, tali vettori risultano essere gli unici ad avere un functional margin pari ad uno, cioè sono quelli che appartengono ad uno dei due iperpiani $w^{\star T}x + b^\star = 1$ o $w^{\star T}x + b^\star = -1$. Tali features sono dette **Support Vector**. Al fine di determinare la misura del margine, sfruttiamo le proprietà dei support vector. Calcoliamo:

$$\begin{aligned}
\|w\|^2 &= \sum_{i,j=1}^{l} y_i y_j \alpha_i^\star \alpha_j^\star \left(x^{(i)}\right)^T x^{(j)} \\
&= \sum_{j \in SV} \alpha_j^\star y_j \sum_{i \in SV} y_i \alpha_i^\star \left(x^{(i)}\right)^T x^{(j)} \\
&= \sum_{j \in SV}^{l} \alpha_j^\star \left(1 - y_j b^\star\right) \\
&= \sum_{j \in SV}^{l} \alpha_j^\star
\end{aligned}$$

dove nell'ultimo passaggio si è sfruttata la condizione $(KKT1.b)$ e SV l'insieme dei support vector. Il margine risulta:

$$\gamma = \frac{1}{\|w\|} = \left(\sum_{j \in SV}^{l} \alpha^\star \right)^{\frac{-1}{2}}$$

Il coefficiente b si ricava attraverso i support vector. Sia (x_s, y_s) un training example appartenente ai support vector. Allora tale coppia deve verificare l'equazione $y_s(w^T x_s + b) = 1$. Sostituendo a w la formula ricavata in (KKT1.w), moltiplicando entrambi i termini dell'equazione per y_s ed infine esplicitando per la variabile b si ricava:

$$b = y_s - \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)} x_s + y_s$$

E' preferibile utilizzare la media dei valori ottenuti dai vari support vector anziché un unico valore:

$$b^\star = \frac{1}{Ns} \sum_{x_s \in SV} \left[y_s - \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)} x_s + y_s \right] \tag{2.9}$$

dove N_s la sua cardinalità.

La hypothesis function risulta quindi:

$$h_{w,b}(w^T x + b) = sgn(w^{\star T} x + b^\star) = sgn\left(\sum_{x_s \in SV}^{m} \alpha_s^\star y_s (x_s)^T x + b^\star \right)$$

Riepilogo

Finora abbiamo definito un SVM in grado di risolvere il problema della classificazione di dati linearmente separabili, nel caso binario. Riepiloghiamo in breve:

- dobbiamo determinare i coefficienti α_i che massimizzano

$$\sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(x^{(i)}\right)^T x^{(j)}$$

sotto il vincolo

$$\alpha_i \geq 0 \quad \forall i \quad e \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Si risolve quindi il problema di massimizzazione tramite un programma per la risoluzione di problemi di programmazione quadratica

- determiniamo i support vector, verificando quali features hanno i corrispettivi moltiplicatori di lagrange > 0;
- calcoliamo

$$w^\star = \sum_{x_s \in SV}^{m} \alpha_s^\star y_s(x_s)$$

- calcoliamo la soglia $b^\star$ attraverso (2.9)
- classifichiamo ogni nuova features x' tramite

$$y' = sgn\left(\sum_{x_s \in SV}^{m} \alpha_s^\star y_s(x_s)^T x + b^\star\right)$$

2.2 Soft Margin

Si è considerato il caso di un data set di dati linearmente separabili. Se applicassimo la soluzione vista ad un set di dati non linearmente separabile non troveremmo nessuna soluzione valida poiché con qualsiasi valore di w e b violerebbe almeno una delle condizioni $y^{(i)}(w^T x^{(i)} + b) \geq 1$. Al fine di risolvere questa tipologia di problemi, si ha quindi la necessità di riformulare il problema di ottimizzazione. Riformuliamo il problema come segue:

$$\begin{array}{cc} \min\limits_{\omega,b,\xi} & \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{m}\xi_i \\ s.t. & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \quad i = 1, \ldots, m \\ & \xi_i \geq 0 \end{array} \tag{2.10}$$

Nella nuova formulazione sono state introdotte le variabili slack, ξ_i, all'interno dei vincoli del problema originale. Tali variabili permetto la misclassificazione delle features di training. Utilizzando tali variabili le condizioni diventano:

$$\begin{array}{ll} w^T x^{(i)} + b \geq 1 - \xi_i & \text{per} \quad y = 1 \\ w^T x^{(i)} + b \leq -1 + \xi_i & \text{per} \quad y = -1 \end{array} \tag{2.11}$$

Nel caso di una errata classificazione basterà assumere per entrambi i casi un valore di $\xi > 0$ per non violare nessun vincolo. Le variabili di slack quindi, permettono di violare alcuni vincoli del problema originale e danno inoltre la possibilità di ottenere un margine geometrico maggiore.

Gli errori di classificazioni compiuti durante la fase di training dell'algoritmo vengono pesati all'interno della funzione costo del problema primale aggiungendo quindi il termine $C\sum_{i=1}^{m}\xi_i$ il cui parametro C assume un significato di trade-off tra errori commessi e valore del margine. Infatti:

- se $C \gg 0$ all'interno della funzione costo l'errore di training ha un peso rilevante. Quindi si andrà a ridurre il numero degli errori di training a discapito del valore del margine, Fig. 2.6;
- se $C \ll 0$ all'interno della funzione costo l'errore di training ha un peso marginale e gli sforzi maggiori saranno compiuti per massimizzare il valore del margine a discapito degli errori di training, Fig. 2.7.

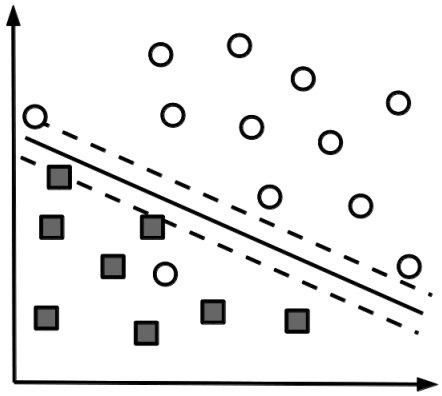


Fig. 2.6: $C \gg 0$

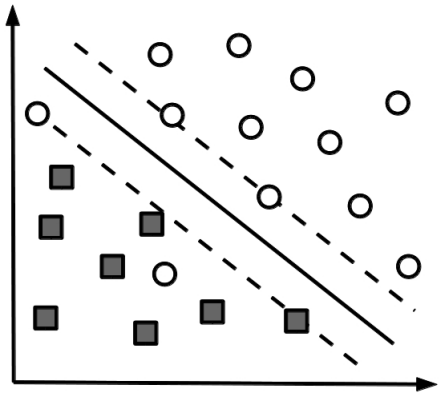


Fig. 2.7: $C \ll 0$

Il valore $\sum_{i=1}^{m} \xi_i$ risulta un upper bound del numero di errori di classificazione dei vettori di training.

Come per il caso precedente ricaviamo il problema duale di più facile soluzione. Per prima cosa ricaviamo la Lagrangiana:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i \left[1 - \xi_{i-} y^{(i)} (w^T x^{(i)} + b) \right] + \sum_{i=1}^{m} \beta_i \left(-\xi_i \right)$$

Siano $(w^\star, b^\star, \xi^\star)$ i valori ottimi del problema primale e siano $(\alpha^\star, \beta^\star)$ i valori ottimi del problema duale, allora tali valori devono verificare le condizione KKT:

- $(KKT1.w) \quad \nabla_w L(w^\star, b^\star, \xi^\star, \alpha^\star, \beta^\star) = 0 \Longrightarrow w^\star = \sum_{i=1}^{m} \alpha_i^\star y^{(i)} x^{(i)}$
- $(KKT1.b) \quad \nabla_b L(w^\star, b^\star, \xi^\star, \alpha^\star, \beta^\star) = 0 \Longrightarrow \sum_{i=1}^{m} \alpha_i^\star y^{(i)} = 0$
- $(KKT1.\xi) \quad \nabla_\xi L(w^\star, b^\star, \xi^\star, \alpha^\star, \beta^\star) = 0 \Longrightarrow C - \alpha_i^\star - \beta_i^\star = 0 \ \ \forall i = 1, \ldots, m$

Utilizzando le condizioni KKT si vuole ottenere una funzione Lagrangiana dipendente solo dai moltiplicatori di Lagrange. Sostituendo le condizioni KKT precedentemente determinate all'interno della Lagrangiana si ottiene:

$$\begin{aligned}
L(w, b, \xi, \alpha, \beta) &= \frac{1}{2} w^T w + \sum_{i=1}^{m} \xi_i \left(C - \alpha_i - \beta_i \right) + \sum_{i=1}^{m} \alpha_i \left[1 - y^{(i)} (w^T x^{(i)}) \right] \\
&= \frac{1}{2} w^T w + \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i y^{(i)} w^T x^{(i)} \\
&= \frac{1}{2} w^T w + \sum_{i=1}^{m} \alpha_i - w^T \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \\
&= \frac{1}{2} w^T w - w^T w + \sum_{i=1}^{m} \alpha_i = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} w^T w \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle x^{(i)}, x^{(j)} \right\rangle = \theta_D(\alpha, \beta)
\end{aligned}$$

Il problema duale assume la forma:

$$\begin{array}{lll} \max\limits_{\alpha,\beta} & \theta_D(\alpha,\beta) & \\ s.t. & \alpha_i \geq 0 & i=1,\dots,m \\ & \beta_i \geq 0 & i=1,\dots,m \\ & \alpha_i + \beta_i = C & i=1,\dots,m \\ & \sum\limits_{i=1}^{m} \alpha_i y^{(i)} = 0 & \end{array}$$

Utilizzando la condizione KKT2 del Teorema 3 riportato in Appendice, possiamo ridurre il numero di vincoli del nostro problema duale. Tale condizione impone che, se $(w^\star, b^\star, \xi^\star)$ è soluzione ottima del problema primale, allora esiste una coppia di valori $(\alpha^\star, \beta^\star)$, soluzione ottima del problema duale, tali da soddisfare le condizioni:

- $(KKT2.a) \quad \alpha_i^\star \left(1 - \xi_i^\star - y^{(i)} \left(w^{\star T} x^{(i)} + b^\star\right)\right) = 0$
- $(KKT2.b) \quad \beta_i^\star \xi_i^\star = 0$

per ogni $i = 1, \dots, m$.
Poiché $\beta_i \geq 0$ dalla (KKT1.ξ) si ricava che il parametro α è compreso tra 0 e C. Più precisamente:

- nel caso in cui $\alpha_i^\star > 0$, da (KKT2.a), si ottiene $1 - \xi_i^\star - y^{(i)} \left(w^{\star T} x^{(i)} + b^\star\right) = 0$. Poiché per ipotesi iniziali $\xi_i \geq 0$, si ha che $y^{(i)} \left(w^{\star T} x^{(i)} + b\right) \leq 1$. Tale vincolo risulta più stringente quando $\beta_i^\star > 0$. In tal caso dalla (KKT2.b) si ricava che $\xi_i^\star = 0$. Si ha quindi che $y^{(i)} \left(w^{\star T} x^{(i)} + b\right) = 1$.

- Se $\alpha_i^\star = 0$, dal terzo vincolo del problema duale si ottiene $\beta_i^\star = C > 0$ e quindi $\xi_i^\star = 0$ da (KKT2.b). Dal primo vincolo del problema primale, (2.10), si ottiene: $y^{(i)} \left(w^{\star T} x^{(i)} + b\right) \geq 1$.

- Infine consideriamo il caso in cui $\alpha_i^\star = C$ questo implica che $\beta_i^\star = 0$ e che $\xi_i^\star > 0$. Quindi dalla (KKT2.a) risulta $y^{(i)} \left(w^{\star T} x^{(i)} + b\right) \leq 1$.

Riassumendo :

$$\begin{array}{rcl} \alpha_i^\star = 0 & \Longrightarrow & y^{(i)} \left(w^{\star T} x^{(i)} + b\right) \geq 1. \\ 0 < \alpha_i^\star < C & \Longrightarrow & y^{(i)} \left(w^{\star T} x^{(i)} + b\right) = 1. \\ \alpha_i^\star = C & \Longrightarrow & y^{(i)} \left(w^{\star T} x^{(i)} + b\right) \leq 1. \end{array}$$

Ora, notando che ogni coppia di vincoli $\beta_i \geq 0$ e $\alpha_i + \beta_i = C$ è equivalente al vincolo $\alpha_i \leq C$ e che $\alpha_i^\star$ è compreso fra 0 e C, possiamo riscrivere il problema duale :

$$\begin{array}{lll} \max\limits_{\alpha,\beta} & \sum\limits_{i=1}^{m} \alpha_i - \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left(x^{(i)}\right)^T x^{(j)} & \\ s.t. & 0 \leq \alpha_i \leq C & \\ & \sum\limits_{i=1}^{m} \alpha_i y^{(i)} = 0 & \end{array} \tag{2.12}$$

Nel problema Duale il parametro C assume il significato di upper bound per i moltiplicatori di Lagrange.

Come definito nel caso precedente, i support vector sono i vettori per i quali il rispettivo moltiplicatore di Lagrange α_i risulta essere diverso da 0. Come fatto per il caso precedente

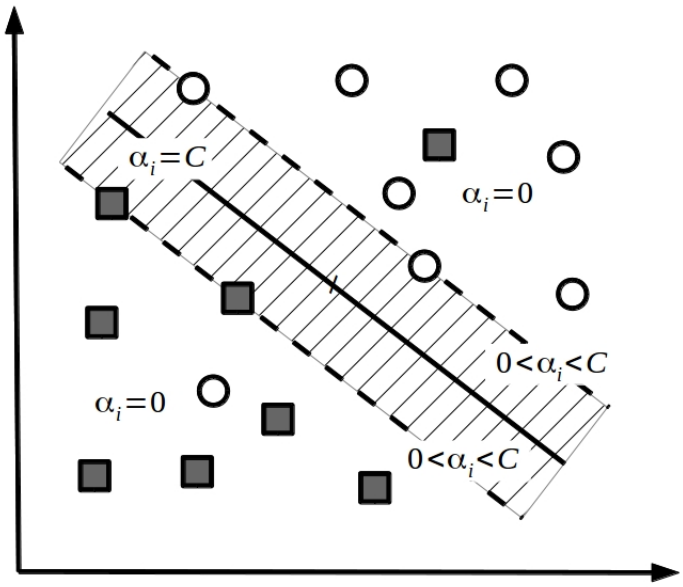


Fig. 2.8: Rappresentazione grafica dei valori α.

andiamo a determinare il valore del margine ottenuto dall'iperpiano ottimo. Sapendo che il margine è dato da $\gamma = \frac{1}{\|w\|}$, calcoliamo:

$$\|w\|^2 = \sum_{i,j=1}^{l} y_i y_j \alpha_i^\star \alpha_j^\star \left(x^{(i)}\right)^T x^{(j)}$$

Essendo C upper bound dei valori di α, maggiore è C maggiore sarà il valore di $\alpha^\star$ e quindi di $\|w\|$. Ciò porta ad una diminuzione del valore del margine dell'iperpiano ottimo.

Riepilogo

Riepiloghiamo quanto fin qui visto per gli insiemi non linearmente separabili:

- si sceglie la penalità da assegnare alle classificazione non corrette di un dato scegliendo un opportuno valore di C.
- si risolvere il problema di minimizzazione quadratica (2.12) al fine di determinare i valori di $\alpha_i^\star$ con $i = 1, \ldots, m$.
- si determinano i support vector, verificando quali features hanno i corrispettivi moltiplicatori di lagrange compresi tra 0 e C
- si calcola
$$w^\star = \sum_{x_s \in Sv}^{m} \alpha_s y_s x_s$$
- si calcola la soglia $b^\star$ attraverso (2.9)
- si classificano le nuove features x' tramite
$$y' = sgn\left(\sum_{x_s \in SV}^{m} \alpha_s^\star y_s (x_s)^T x + b^\star\right)$$

2.3 Kernel Trick

Consideriamo un training set di dati non linearmente separabili. Si vuole trovare un metodo più performante rispetto a quello lineare con slack per suddividere le due classi.
Se si analizzano le forme dei problemi precedenti si può notare che i dati compaiono nella forma di prodotti interni. L'idea è quella di lavorare in uno spazio diverso dallo spazio di input, in cui i dati diventino linearmente separabili. A tal fine definiamo con il termine *features mapping* una mappa non lineare $\phi : \mathbb{R}^n \longrightarrow F : x \longrightarrow \phi(x)$ dove F, il future space, è uno spazio Euclideo di dimensione maggiore di n (eventualmente infinita), cioè uno spazio lineare con un prodotto scalare.

Si considerino ora i trasformati $\phi(x^{(i)})$ del feature set e il problema della determinazione del piano ottimo nel feature space F, in cui i vettori trasformati sono linearmente separabili.

Si ottiene quindi il corrispondente problema duale :

$$\begin{aligned} \max_{\alpha,\beta} \quad & \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(\phi(x^{(i)})\right)^T\phi\left(x^{(j)}\right) \\ s.t. \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{m}\alpha_i y^{(i)} = 0 \end{aligned} \tag{2.13}$$

Si noti che ciò che interessa conoscere al fine della risoluzione del problema duale non è il singolo valore di $\phi(x^{(i)})$ bensì il prodotto interno $\langle\phi(x), \phi(z)\rangle = \phi(x)^T\phi(z)$ che successivamente indicheremo anche con $K_{ij} = K\left(x^{(i)}, x^{(j)}\right)$. Il problema (2.13) lo si può vedere come un problema di SVM lineare ma nello spazio F. Da tale problema saremo in grado di ricavare i moltiplicatori di Lagrange e i support vector. Affinché $\alpha^\star$ sia il valore ottimo della funzione costo è sufficiente che siano rispettate le condizioni KKT, che rimangono immutate rispetto al precedente problema:

$$\begin{aligned} \alpha_i^\star = 0 \quad &\Longrightarrow \quad y^{(i)}\left(w^{\star T}\phi(x^{(i)}) + b\right) \geq 1. \\ 0 < \alpha_i^\star < C \quad &\Longrightarrow \quad y^{(i)}\left(w^{\star T}\phi(x^{(i)}) + b\right) = 1. \\ \alpha_i^\star = C \quad &\Longrightarrow \quad y^{(i)}\left(w^{\star T}\phi(x^{(i)}) + b\right) \leq 1. \end{aligned}$$

dove $w^\star = \sum_{x_s \in SV} \alpha_s^\star y_s \phi(x_s)$. Quindi, sostituendo $w^\star$ in $\left(w^{\star T}\phi(x^{(i)}) + b\right)$ si ricava:

$$w^{\star T}\phi(x^{(i)}) + b = \sum_{x_s \in SV} \alpha_s^\star y^s \phi(x_i)^T\phi(x) + b^\star = \sum_{x_s \in SV} \alpha_s^\star y^s K(x_s, x) + b^\star$$

e

$$b^\star = \frac{1}{Ns}\sum_{x_s \in SV}\left[y_s - \sum_{i=1}^{m}\alpha^{(i)}y^{(i)}K(x^{(i)}, x_s) + y_s\right]$$

Si noti che nella formulazione del problema duale (2.13) nel calcolo di $w^\star$ e di $b^\star$ non è necessaria la conoscenza esplicita dell'operatore $\phi(\bullet)$, bensì basta conoscere il prodotto scalere tra le features. Dati due vettori x' e x'' definiremo con Kernel, K_{ij} il prodotto scalare dei loro trasformati:

$$K_{ij} = \phi(x')^T\phi(x'')$$

Essenziale per il prosieguo della discussione risulta il seguente teorema.

Teorema 1. *(Teorema di Mercer) Ogni matrice K semidefinita positiva e simmetrica è una matrice kernel, per la quale esiste una funzione $\phi(x)$ tale che:*

$$K_{ij} = K\left(x^{(i)}, x^{(j)}\right) \quad \forall \quad i, j = 1, \ldots, m$$

Esistono quindi in letteratura diverse tipologie di kernel. La più utilizzata è la tipologia di kernel RBF. Nelle sezioni successive vengono riportati i Kernel che verranno poi utilizzati nelle fasi sperimentali.

Kernel Polinomiale Il kernel polinomiale è così definito:

$$K\left(x^{(i)}, x^{(j)}\right) = \left(\gamma x^{(i)T} x^{(j)} + r\right)^d \tag{2.14}$$

dove γ e r sono dei parametri mentre d è il grado del kernel. Il parametro γ è un indice della dispersione dei dati nel feature space.
Il polinomial kernel è quindi il prodotto scalare dei due vettori $\phi(x^{(i)})$ e $\phi(x^{(j)})$.

Nel caso in cui d=2 tali vettori risultano avere la forma :

$$\phi(x) = \left[r, \sqrt{\gamma r}x_1, \ldots, \sqrt{\gamma r}x_n, \gamma x_1^2, \ldots, \gamma x_n^2, \gamma x_1 x_2, \ldots, \gamma x_{n-1} x_n\right]$$

Per un kernel polinomiale la dimensione di $\phi(x)$ è:

$$C(n+d, d) = \frac{(n+d)!}{d!n!}$$

Consideriamo il kernel polinomiale $k\left(x^{(i)}, x^{(j)}\right) = \left(\gamma x^{(i)T} x^{(j)}\right)^2$ dove le features hanno dimensione 1 . Allora le $\phi(x)$ avranno dimensione 3.

Kernel RBF: Radial Basis Function Il kernel RBF è così definito:

$$K_{RBF}(x, z) = \exp\left(-\gamma d_{RBF}(x, z)\right) = \exp\left(-\gamma \|x - z\|^2\right)$$

con $\sigma > 0$.
Tale kernel può essere pensato come un indice di similarità perché nel caso :

- $x \simeq z$ allora $K(x, z) \simeq 1$
- $x \neq z$ allora $K(x, z) \simeq 0$

dove γ gioca un ruolo di selettività nel nostro problema, ossia è l'indice con cui stabiliamo la similarità dei dati:

- ad un γ grande corrisponde una curva gaussiana stretta/selettiva. Quindi basta un piccolo spostamento delle features da uno dei vettori SV per avere un $K(x, z) \simeq 0$
- ad un γ piccolo corrisponde una curva gaussiana poco selettiva. Quindi $K(x, z)$ tende a zero più lentamente all'aumentare della distanza della features dal SV.

Si può pensare a γ come all'inverso del raggio di una sfera con centro posto sui SV, dove tutti i punti interni a questa sfera possono essere considerati simili al SV. Più γ è piccolo, maggiore è il raggio della sfera e maggiore è il numero di vettori che potenzialmente possono essere considerati simili al SV. Con γ grande, invece, tale raggio risulterà più piccolo. La scelta di tale parametro all'interno del nostro algoritmo risulta molto importante e può tener conto di diversi fattori come ad esempio il rumore delle feature.

Kernel χ^2 Tale Kernel è definito dalla funzione:

$$K_{\chi^2}(x, z) = \exp\left(-\gamma d_{\chi^2}(x, z)\right) = \exp\left(-\gamma \sum_i \frac{(x_i - z_i)^2}{x_i + z_i}\right)$$

Come per il caso del kernel precedente il parametro γ risulta un indice di similarità dei dati.

Kernel L_1 Tale Kernel sarà indicato con la sigla L1.

$$K_{L_1}(x,z) = \exp\left(-\gamma d_{L_1}(x,z)\right) = \exp\left(-\gamma \sum_i |x_i - z_i|\right)$$

Kernel Histogram Intersection Kerne Tale Kernel sarà indicato con la sigla HIK.

$$K_{HIK}(x,z) = \sum_i min(x_i, z_i)$$

Il Kernel RBF,L_1 e χ^2 sono kernel esponenziali dove γ ha lo stesso effetto descritto per il kernel RBF. La differenza tra questi tre kernel è la metrica con cui si misura la distanza tra le features.

Capitolo 3

Primo Problema

In questo capitolo andremo ad approfondire la struttura dell'algoritmo implementato per il primo problema e presenteremo i test svolti. Risulta inizialmente necessario andare a definire, più nel dettaglio, alcune terminologie utilizzate nel primo capitolo al fine di declinarle all'interno del nostro problema. Nella prima parte del capitolo quindi verrà definita la differenza tra candidato e difetto al fine di comprendere meglio i risultati della fase di test. Nella seconda parte verranno presentati le features utilizzate in ingresso ai classificatori SVM e gli schemi del classificatore. Infine saranno riportati i test svolti con l'algoritmo di classificazione.

3.1 Definizione di Difetto e Candidato

Nel primo capitolo si è parlato più volte dei candidati. In questo problema, con il termine candidato, c_n, indichiamo una terna di elementi costituita da una matrice e una coppia di vettori $(BW_{c_n}, Pint_{c_n}, Pext_{c_n})$. La matrice BW_{c_n} ha dimensioni $w_{c_n} \times h_{c_n}$, ossia la dimensione del bounding box (la più piccola regione rettangolare contenente un blob, cioè una connessione di pixel di almeno 40 elementi). Tale matrice è un array logico dove i pixel di valore 1 rappresentano il blob mentre i pixel di valore 0 rappresentano il background del blob. I due vettori restituiscono i valori dei pixel dell'immagine normalizzata che appartengono al blob, $Pint_{c_n}$ o al suo background $Pext_{c_n}$:

$$\begin{aligned} BW(i,j)_{c_n} &= \begin{cases} 1 & \text{se} \quad (i,j) \in \text{blob} \\ 0 & \text{se} \quad (i,j) \in \text{background} \end{cases} \\ 0 \leq Pint_{c_n}(i) &\leq 1 \quad \text{con} \quad 1 \leq i \leq N_{int} \\ 0 \leq Pext_{c_n}(i) &\leq 1 \quad \text{con} \quad 1 \leq i \leq N_{ext} \end{aligned} \tag{3.1}$$

dove N_{int} e N_{ext} sono la cardinalità dei vettori $Pint$ e $Pext$.

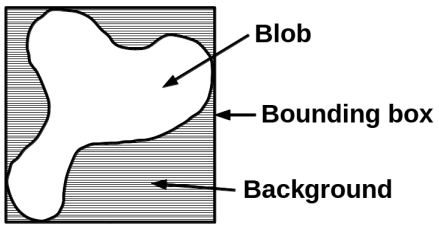


Fig. 3.1: Esempio di blob, background e baunding box.

Per la comprensione delle successive fasi è molto importante definire cosa si intende per *classificazione corretta*, *falso positivo* e *falso negativo*.

Finora abbiamo parlato solo di candidati. L'obiettivo finale però è quello di determinare, classificare e individuare i difetti presenti nell'immagine. Se si fa ora riferimento all'immagine Fig. 1.6, si nota come ciò che a prima vista può sembrare un unico difetto, venga invece suddiviso in quattro diversi candidati. Quindi sorge spontaneo chiedersi, se per poter affermare di aver individuato e classificato correttamente un difetto, sia necessario aver classificato come difetto tutti e quattro i candidati oppure se sia sufficiente la corretta classificazione anche solo di uno di questi candidati. Per chiarire le scelte fatte a questo proposito e dare una descrizione rigorosa del metodo utilizzato, è opportuno chiarire il significato di alcune notazioni, che riportiamo di seguito.

- $I^{(i)}$: i-esima immagine del dataset;
- $C^{(i)}$: insieme dei candidati dell'immagine $I^{(i)}$; $\beta^{(i)}$ è la sua cardinalità.
- $N^{(i)}$: il numero dei difetti nell'immagine $I^{(i)}$;
- $N_j^{(i)} = \{n_1^{(i)}, n_2^{(i)}, \ldots\}$: insieme dei candidati che rappresentano il difetto j; $\aleph_j^{(i)}$ è la cardinalità di tale insieme
- $M^{(i)} = \{m_1^{(i)}, m_2^{(i)}, \ldots\}$: insieme dei candidati che non individuano nessun difetto di $I^{(i)}$; $\theta^{(i)}$ è la cardinalità di tale insieme
- L'insieme $C^{(i)}$ è dato quindi dall'unione :

$$C^{(i)} = M^{(i)} \cup N_1^{(i)} \cup N_2^{(i)} \cup \ldots \cup N_{N^{(i)}}^{(i)}$$

  e la sua cardinalità è:

$$\beta^{(i)} = \theta^{(i)} + \aleph_1^{(i)} + \aleph_2^{(i)} + \ldots + \aleph_{N^{(i)}}^{(i)}$$

- siano $\left(x_j^{(i)}, y_j^{(i)}\right)$ la coppia composta da feature e label in ingresso al classificatore. Tale coppia rappresenta un elemento dell'insieme dei candidati $C^{(i)}$.
- $\widehat{y}_j^{(i)}$: output del classificatore rispetto alla coppia di input $\left(x_j^{(i)}, y_j^{(i)}\right)$.

Allora si ha :

Classificazione corretta (CC) se **almeno uno** degli $N_j^{(i)}$ candidati che rappresentano il difetto j viene classificato correttamente dal classificatore, cioè se

$$\exists x_j^{(i)} \in N_j^{(i)} : \widehat{y}_j^{(i)}\left(x_j^{(i)}\right) = y_j^{(i)}\left(x_j^{(i)}\right)$$

Falso negativo (FN) se **nessuno** degli $N_j^{(i)}$ candidati che rappresentano il difetto j viene classificato correttamente, cioè se

$$\forall x_j^{(i)} \in N_j^{(i)} : \widehat{y}_j^{(i)}\left(x_j^{(i)}\right) \neq y_j^{(i)}\left(x_j^{(i)}\right)$$

Falso positivo (FP) se **uno** dei candidati che non rappresenta nessuno degli $N^{(i)}$ difetti dell'immagine $I^{(i)}$ viene riconosciuto come difetto, cioè se:

$$\exists x_j^{(i)} \in M_j^{(i)} : \widehat{y}_j^{(i)}\left(x_j^{(i)}\right) \neq y_j^{(i)}\left(x_j^{(i)}\right)$$

Vero positivo (VP) se uno degli $N_j^{(i)}$ candidati che rappresentano il difetto j viene classificato correttamente dal classificatore, cioè se:

$$\forall x_j^{(i)} \in N_j^{(i)} : \widehat{y}_j^{(i)}\left(x_j^{(i)}\right) = y_j^{(i)}\left(x_j^{(i)}\right)$$

Queste definizioni sono da tener ben presenti nella comprensione dei risultati nella fase di test. Tali definizioni determinano delle possibili casistiche durante il confronto tra due diversi classificatori:

- può capitare che ad un pari numero di veri positivi corrisponda un diverso numero di classificazioni corrette. Ad esempio, consideriamo quattro difetti presenti in un'immagine, ciascuno individuato da quattro candidati. Supponiamo che, effettuata la classificazione, si ottengano quattro veri positivi. Ci sono dunque ben quattro situazioni diverse che si possono verificare: la prima è che tutti i veri positivi individuino lo stesso difetto, ottenendo una sola classificazione corretta. Gli altri tre casi rappresentano rispettivamente le situazioni in cui si ha la classificazione corretta di due, tre o quattro difetti.
- per tale motivo, l'aumento dei veri positivi non comporta necessariamente l'aumento delle classificazioni corrette.

Tali osservazioni evidenziano il fatto che analizzare i classificatori basandosi anche o solo su i veri positivi può risultare fuorviante. Inoltre sottolineano la difficoltà di comprensione degli andamenti dei vari classificatori al variare dei parametri.

3.2 Features

Le features sono un insieme di proprietà qualitative che permettono di caratterizzare il candidato. L'idea è quella di determinare una caratteristica comune a tutti i candidati. Tale individuazione risulta difficile per questo problema in quanto i difetti e i candidati sono tra loro molto diversi. Si è scelto quindi di dividere i candidati per forma geometrica e basare la classificazione difetto/non difetto sul colore di tali candidati. Le features quindi andranno a misurare questi due aspetti: forma e colore del candidato.
Definiamo ora le features che saranno utilizzate in seguito.

3.2.1 Geometric Features

I descrittori geometrici determinati e utilizzati attraverso la funzione *regionprops* di matlab sono:

- *Area*: numero di pixel che formano il blob
- *Perimetro*: calcolato come la distanza tra ciascuna coppia adiacente di pixel appartenente al bordo del blob
- *Asse maggiore*: lunghezza (in pixel) dell'asse maggiore dell'ellisse che possiede i medesimi momenti centrali normalizzati del secondo ordine del blob.
- *Asse minore*: lunghezza (in pixel) dell'asse minore dell'ellisse che possiede i medesimi momenti centrali normalizzati del secondo ordine del blob.

3.2.2 Color Fetaures

Le color features sono una delle tipologie dei descrittori più utilizzate nei problemi di classificazioni di immagini. All'interno del progetto si sono adoperate due tipologie di color features: i parametri statistici del colore, come media e varianza e gli istogrammi di colore. Di seguito riportiamo una descrizione di tali descrittori:

Media pixel interni media dell'intensità dei pixel del blob

$$MEAN_{int} = \frac{1}{N_{int}} \sum_{p_i \in Pint_{cn}} p_i$$

dove N_{int} è la lunghezza del vettore $Pint_{cn}$.

Varianza pixel interni La varianza dell'intensità dei pixel del blob

$$VAR_{int} = \frac{1}{N_{int}} \sum_{p_i \in Pint_{cn}} (p_i - MEAN_{int})^2$$

Media pixel esterni media dell'intensità dei pixel esterni al blob:

$$MEAN_{ext} = \frac{1}{N_{ext}} \sum_{p_i \in Pext_{cn}} p_i$$

dove N_{ext} è la lunghezza del vettore $Pint_{cn}$.

Varianza pixel interni La varianza dell'intensità dei pixel esterni al blob:

$$VAR_{ext} = \frac{1}{N_{ext}} \sum_{p_i \in Pext_{cn}} (p_i - MEAN_{ext})^2$$

istogramma dei livelli di grigio Sia U un vettore dell'intensità dei pixel di un'immagine in scala di grigio, l'istogramma è una mappa, che mappa i valori di U in un vettore $H = [H_0, H_1, \ldots, H_n]$, dove n indica il numero dei bins e

$$H_i = Card\left\{u \in U | i\frac{G}{n} \leq u < (i+1)\frac{G}{n}\right\}$$

Nel nostro caso il vettore U è dato dall'unione di due vettori $Pint_{c_n} \cup Pext_{c_n}$. Poiché l'area del bounding box è variabile, risulta necessario normalizzare i valori dell'istogramma. Ossia l'istogramma utilizzato sarà:

$$H_{norm} = \left[\frac{H_1}{\sum_{i=1}^{n} H_i}, \frac{H_2}{\sum_{i=1}^{n} H_i}, \ldots, \frac{H_n}{\sum_{i=1}^{n} H_i}\right]$$

Nella fase di test si andrà a variare il valore di n.

3.3 Struttura del classificatore

La prima domanda a cui si è cercato di dare una risposta è come strutturare il nostro classificatore. La varietà di tipologie dei difetti suggerisce di non cercare di creare un unico classificatore. Da un osservazione delle immagini e dei difetti si è deciso di raggruppare i difetti in base alla loro forma geometrica. Tale scelta è stata compiuta in quanto volendo utilizzare il colore come caratteristica per la classificazione difetto/non difetto, si è notato che i difetti lunghi, come i trucioli o le sporgenze, sono caratterizzati da un intensità elevata di gradazione di grigio, al contrario dell'acqua o dei difetti ad S la cui intensità risulta più attenuata.
Le classi di forma scelte per dividere i candidati sono tre: *lunghi e stretti* (ad esempio i trucioli o altri tipi di scarti), *difetti circolari* (come le gocce di acqua) e la classe che chiameremo *altro*, che comprende tutte quelle forme che non fanno parte né della prima, né della seconda classe.
Lo schema adottato è riportato in Figura 3.2. Ogni candidato, una volta classificato per forma, subisce una seconda classificatoria difetto/non difetto da uno specifico classificatore. Per una lettura più scorrevole delle fasi successive si indicherà con classificatore tutta la struttura di classificazione di Figura 3.2, con i termini di sottoclassificatori di forma i classificatori evidenziati di grigio e con i termini di sottoclassificatori di difetti i classificatori evidenziati in colore celeste, classificatori difetto/non difetto.

	Lunghi	Circolari
# **Immagini**	13	8
# **Candidati**	42	28
# **Positive Labels**	21	5

Tabella 3.1: Database training dei classificatori di forma

	Lunghi	Circolari
C	4	1
γ	12,6	8,4
grado polinomio	3	3

Tabella 3.2: Parametri con cui sono stati allenati i sottoclassificatori di forma, per i quali è stato utilizzato un kernel polinomiale.

3.3.1 Sotto-classificatori di forma

Tali sotto-classificatori sono stati allenati sfruttando le geometric features e utilizzando un Kernel polinomiale. I dati del dataset utilizzato nella fase di training dei due classificatori e i parametri dei classificatori sono riportati nelle tabelle 3.1 e 3.2. Tali classificatori non vengono mai modificati, ossia i modelli di classificatori così ottenuti rimangono invariati per tutte le future fasi di training e per la fasi di test.

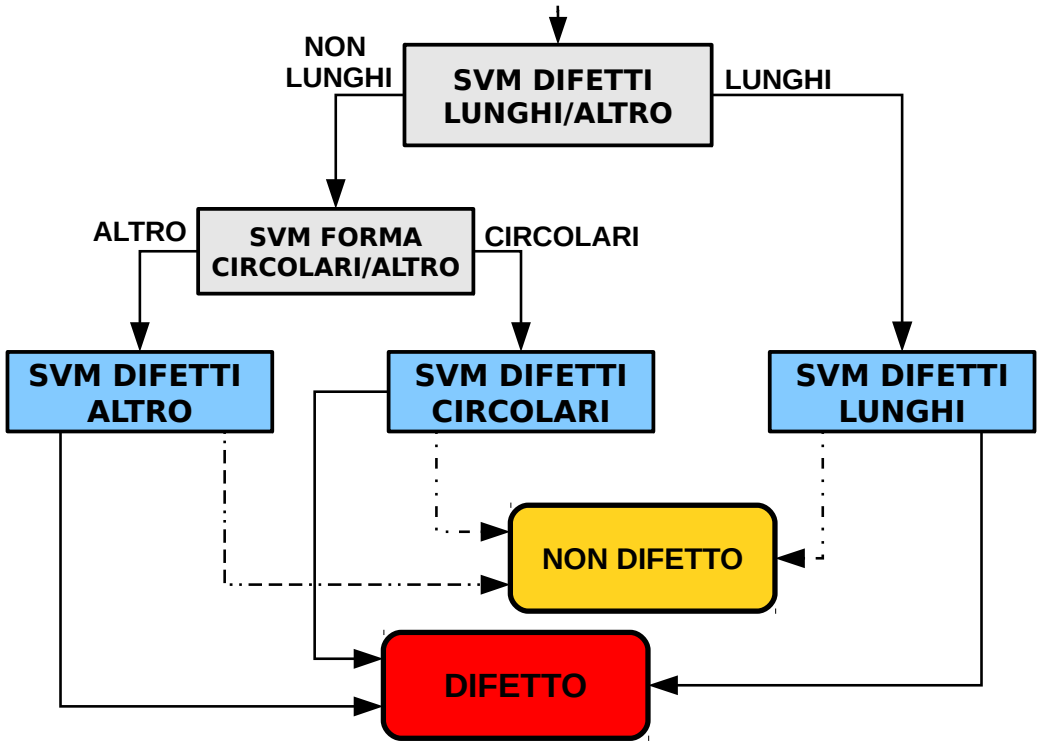


Fig. 3.2: Primo schema di classificazione; in grigio sono indicati i sottoclassificatori di forma e in celeste i sottoclassificatori di difetti

3.3.2 Sotto-classificatori di difetti

Il database di immagini fornito da Video Systems presenta un elevato numero di immagini, circa 3500 per ciascuna tipologia. Tra queste però, il numero di immagini con difetti risulta intorno al 5%. Si è formato un dataset iniziale di immagini, contenente tutte le immagini con difetti ed un insieme di immagini casuali, che non presentavano difetti. Da esso si sono ricavati quattro database: due database di training (che variano uno dall'altro per cinque immagini) e due database di test, uno per ogni tipologia. Si è cercato di dare a tali insiemi una cardinalità necessaria a garantire un buon training dei classificatori e un numero sostanzioso di immagini per la fase di

test.
I sottoclassificatori di difetti rappresentano il punto critico della nostra tesi. Nei database di training, formati in maniera sperimentale, sono state inserite tutte le tipologie di difetti precedentemente descritte e sono presenti immagini di entrambe le tipologie, REV e STD. La composizione di tali database è riportata in tabella 3.3. In essa sono riportati anche i numeri di candidati divisi per forma. Si può notare come i difetti circolari risultino la forma con minor numero di candidati.
Le fasi logiche con cui si svolge il training dei sotto-classificatori di difetto risultano:

- creazione del database di training;
- classificazione manuale dei candidati: ogni candidato ottenuto dal database di training viene assegnato alla classe di appartenenza: positive class, ossia difetto, o negative class, non difetto (training examples);
- ove necessario ogni features estratta dai candidati viene riscalata in un range tra 0 e 1;
- ogni candidato viene classificato in base alla forma tramite i sottoclassificatori di forma. In questo modo vengono a crearsi tre nuovi dataset di training: dataset difetti lunghi, dataset difetti circolari e dataset difetti altro. Tali dataset saranno utilizzati per allenare i sottoclassificatori di difetti.
- ogni sottoclassificatore di difetto viene allenato impostando i valori di C e i paramentri del kernel utilizzato.

La fase di riscalatura nei problemi di classificazione tramite SVM, può assumere in apparenza un ruolo di secondo piano. Tale fase è esplicitamente richiesta nell'utilizzo delle funzioni della libreria *libSVM* al fine di garantire una stabilità computazionale. Nell'esecuzione dei test si è notato però, che la riscalatura dei vettori di features è in grado di influenzare in maniera non banale i risultati di classificazione. Più i valori impostati di massimo (M) e minimo (m) utilizzati nella funzione di riscalatura riportata in (3.2), garantiscono una distribuzione omogenea all'interno dell'intervallo, migliori risulteranno i risultati finali.

$$fv_{rescale} = \frac{fv - m}{M - m} \tag{3.2}$$

		DAT A	**DAT B**
# **Immagini**		34	39
# **Difetti**		26	30
CAND.	**LUNGHI**	99(20)	105(25)
	CIRCOLARI	16(8)	16(8)
	ALTRO	130(13)	143(21)
	TOTALE	245(41)	264(44)

Tabella 3.3: Composizione dei dataset di training utilizzati per allenare i sottoclassificatori di difetto. Tra parentesi sono riportati il numero dei candidati appartenenti alla classe difetto

3.4 Indici Valutazione e Database Di Test

Al fine di misurare la bontà di un classificatore, per ogni simulazione si sono raccolti alcuni indici globali del classificatore o dei singoli sottoclassificatori di difetti. Nella tabella 3.4 riportiamo i valori raccolti e il loro significato. Si noti che per ogni sottoclassificatore di difetto sono stati raccolti i veri positivi e non le classificazioni corrette poiché andare a determinare il numero

Indici	
	TOTALE
FP	numero totale di falsi positivi (viene indicata la percentuale rispetto al totale dei candidati appartenenti alla classe non difetto)
FN	numero totale di falsi negativi
CC	numero di difetti classificati correttamente
Sensitività	percentuale di difetti presenti nel database di test classificati correttamente
Precisione	probabilità che un positivo ritornato dal classificatore sia corretto
	SOTTOCLASSIFICATORE DI DIFETTI LUNGHI
FP	numero di falsi positivi restituiti dal sottoclassificatore dei difetti lunghi (viene indicata la percentuale rispetto al totale dei candidati appartenenti alla classe non difetto)
FN	numero di falsi negativi restituiti dal sottoclassificatore di difetti lunghi
VP	numero di veri positivi restituito dal sottoclassificatore dei difetti lunghi
	SOTTOCLASSIFICATORE DI DIFETTI CIRCOLARI
FP	numero di falsi positivi restituiti dal sottoclassificatore dei difetti circolari (viene indicata la percentuale rispetto al totale dei candidati appartenenti alla classe non difetto)
FN	numero di falsi negativi restituiti dal sottoclassificatore di difetti circolari
VP	numero di veri positivi restituito dal sottoclassificatore dei difetti circolari
	SOTTOCLASSIFICATORE DI DIFETTI ALTRO
FP	numero di falsi positivi restituiti dal sottoclassificatore dei difetti altro (viene indicata la percentuale rispetto al totale dei candidati appartenenti alla classe non difetto)
FN	numero di falsi negativi restituiti dal sottoclassificatore di difetti altro
VP	numero di veri positivi restituito dal sottoclassificatore dei difetti altro

Tabella 3.4: Spiegazione degli indici presenti nelle tabelle di raccolta dati

di classificazioni corrette dei sottoclassificatori non avrebbe aggiunto alcuna informazione alla nostra analisi.

Nell'analisi delle prestazioni dei singoli classificatori si farò riferimento in particolare a due indici:

Sensitività percentuale di difetti presenti nel database di test classificati correttamente

$$\textbf{Sensitività} = \frac{CC}{CC + FN}$$

Precisione percentuale di candidati classificati come positivi che effettivamente lo sono

$$\textbf{Precisione} = \frac{CC}{CC + FP}$$

Come detto in precedenza dalle immagini fornite dal Video Systems si sono ricavati due dataset di immagini per la fase di test, uno per ciascuna tipologia. I dati di tale dataset sono stati riportati in tabella 3.5. Rispetto a tali tabelle si possono compiere alcune osservazioni:

- per ogni immagine si hanno in media 10 candidati;
- per la tipologia REV i candidati appartenenti alla classe difetto sono l'8%, mentre per la seconda tipologia sono il 44%. Tale disparità ha due ragioni. La prima dipende dalla

diversa tipologia del fondo. Infatti dal fondo rugoso che caratterizza REV si ricavano un maggior numero di candidati appartenenti alla classe non difetto rispetto alla tipologia STD. La seconda ragione è la tipologia dei difetti. Negli oggetti STD il difetto che si presenta con più frequenza è la presenza di liquido. Tale difetto ha un notevole numero di candidati che lo individuano rispetto ad altre tipologie di difetti come i trucioli, per due ragioni: la prima è che il liquido riflette parzialmente la luce e ciò porta nel suo intorno ad un aumento del numero di candidati; la seconda ragione riguarda la sua dimensione che risulta molto più grande rispetto ad altre tipologie di difetti.

- come si era visto per il database di training i candidati circolari risultano l'insieme con cardinalità minore
- l'insieme altro risulta per entrambe le tipologie l'insieme con maggioro cardinalità. Mentre per la tipologia STD la gran parte dei difetti sono rappresentati da candidati di tale forma, per la tipologia REV il numero maggiore di candidati della classe difetti risulta di forma lunga.

	REV		**STD**	
	TOT	**positivi**	**TOT**	**positivi**
# **immagini**	210		234	
# **difetti**	92		128	
# **candidati**	2166	176(8,1%)	1621	726(44,8%)
# **candidati lunghi**	659	91(13,8%)	448	233(52,0%)
# **candidati circolari**	100	12(12,0%)	188	99(52,6%)
# **candidati altro**	1407	73(5,2%)	985	394(40,0%)

Tabella 3.5: Dati di composizione dei database di Test per le due tipologie REV e STD

3.5 Classificatore Video Systems

Video Systems aveva sviluppato un suo classificatore raccogliendo alcune statistiche del colore e della forma geometrica, come l'eccentricità, di un insieme di candidati appartenenti a dei difetti. Queste statistiche sono state utilizzate come vincoli per classificare nuovi candidati. Per poter effettuare un confronto con l'algoritmo implementato da Video Systems si sono raccolti i risultati di tale classificatore su i dataset di test precedentemente presentati. I risultati sono stati riportati in tabella 3.6.

	REV	STD
FP	162(8,14%)	168(26,40%)
FN	59	29
CC	33	99
Sensitività	35,87%	77,34%
Precisione	16,90%	37,07%

Tabella 3.6: Risultati dell'algoritmo di classificazione di Video Systems

Dalla tabella si nota un valore molto basso della precisione e un comportamento molto differente della sensitività in relazione alla tipologia di oggetto considerata. Infatti, mentre per STD si ha un buon valore di sensitività (al di sopra del 75%), per REV il valore della sensitività risulta non accettabile.

3.6 Test e Risultati

In tale sezione verranno riportati test effettuati e le motivazioni per le quali sono stati effettuati. É importante inoltre sottolineare che lo scopo di tali test è quello di cercare di ottenere un valore di sensitività il più elevato possibile, cercando di mantenere la precisione ad un valore almeno del 60%. All'inizio di ogni test verranno specificati la tipologia del vettore di features, il database di training, i Kernel e lo schema di classificazione utilizzato. In questa fase vengono riportate solo le tabelle dei risultati utili alla comprensione del test e delle conclusioni tratte.

3.6.1 Primo Test

Tipologia di Features: Come tipologia di features in questo primo test si è utilizzato un vettore di dimensione 1×4 formato dalla media e dalla varianza dell'intensità del colore blob e del background contenuto nel bounding box. Il vettore delle features è così composto:

$$fv = [MEAN_{int} \quad VAR_{int} \quad MEAN_{ext} \quad VAR_{ext}]$$

Database Training: Database di training A

Tipologia di Kernel utilizzato: Kernel RBF

Schema di classificazione: Schema 1

In questa prima test sono stati testati 4 classificatori con i seguenti parametri:

- **CL1**: Kernel RBF con parametri $\gamma = 0.02$ e C $=2^{15}$
- **CL2**: Kernel RBF con parametri $\gamma = 0.9$ e C $=2^{15}$
- **CL3**: Kernel RBF con parametri $\gamma = 128$ e C $=2^{3}$
- **CL4**: Kernel RBF con parametri $\gamma = 512$ e C $=2^{3}$

Con tale test si vogliono ottenere due informazioni:

- confrontare i risultati del classificatore implementato con quello di Video Systems al fine di capire quale classificatore ha prestazioni migliori,
- cercare di capire quali valori dei parametri C e γ, risultino i più adatti al nostro obiettivo, cioè ottenere la sensitività maggiore con un valore di precisione almeno del 60%.

Come detto nel capitolo precedente un valore di γ piccolo comporta poca selettività con il rischio di ottenere molte misclassificazioni. Al fine di ridurre gli errori di classificazione si avrà bisogno di aumentare il valore di C. All'aumentare di γ e quindi della selettività del kernel RBF si avrà sempre meno bisogno di andare ad utilizzare un valore di C elevato poiché è lo stesso γ a garantire un piccolo numero di misclassificazioni. Con i classificatori CL2 e CL3 quindi si vuole capire se, al fine di raggiungere il nostro scopo, risulti più conveniente la prima situazione, ossia poca selettività, o un valore di γ più selettivo. I valori di C associati invece sono stati scelti per ottenere un valore della sensitività nella fase di training che sia almeno dell'80%. Mentre con i classificatori CL1 e CL4 si è voluto estremizzate ancor di più la selettività del nostro classificatore a parità del parametro C.

In tabella 3.7 sono stati riportati i risultati ottenuti nella fase di test dove sono evidenziati in grigio i classificatori che presentano il più alto valore di sensitività. Se si osservano le Figure 3.3 e 3.4, si nota la netta differenza tra il numero di falsi positivi dei quattro classificatori testati rispetto al classificatore Viedeo Systems (VS). Tale risultato può essere considerato il primo notevole miglioramento prestazionale. Diversa invece risulta l'efficacia dei classificatori

		REV				STD			
		CL 1	**CL 2**	**CL 3**	**CL 4**	**CL 1**	**CL 2**	**CL 3**	**CL 4**
GLOBALE	**FP**	18(0,90%)	55(2,76%)	42(2,11%)	32(1,61%)	16(1,79%)	27(3,02%)	17(1,90%)	12(1,34%)
	FN	33	24	27	40	53	37	42	83
	CC	59	68	65	52	75	91	86	45
	# Dif.	92	92	92	92	128	128	128	128
	Sens.	64,13%	73,91%	70,65%	56,52%	58,59%	71,09%	67,19%	35,16%
	Prec.	76,62%	55,28%	60,75%	61,90%	82,42%	77,12%	83,50%	78,95%
LUNGHI	**FP**	4(0,70%)	13(2,29%)	15(2,64%)	18(3,17%)	2(0,93%)	4(1,86%)	1(0,47%)	0(0,00%)
	FN	47	43	49	55	198	182	206	221
	VP	44	48	42	36	35	51	27	12
CIRC.	**FP**	7(7,95%)	16(18,18%)	2(2,27%)	1(1,14%)	9(10,11%)	13(14,61%)	6(6,74%)	4(4,49%)
	FN	1	0	3	6	49	29	60	69
	VP	11	12	9	6	49	69	39	30
ALTRO	**FP**	7(0,52%)	26(1,95%)	25(1,87%)	13(0,97%)	5(0,85%)	10(1,69%)	10(1,69%)	8(1,35%)
	FN	36	29	30	51	287	272	275	362
	VP	37	44	43	22	108	123	119	32

Tabella 3.7: Risultati primo test

per quanto riguarda il numero di classificazioni corrette. Mentre per la tipologia REV si riesce ad ottenere un valore di sensitività quasi doppio rispetto a quella del classificatore VS, con il classificatore CL2 per le immagini STD, non si riescono ad uguagliare le prestazioni di VS.
Cerchiamo ora di analizzare i dati dei classificatori testati.
In Figura 4.6 e 3.6 sono stati riportati gli istogrammi dei falsi positivi e dei falsi negativi di ciascun classificatore, divisi per forma del candidato. Dalla Figura 3.6 si denota la diversità nel numero di candidati che compongono un difetto tra le due tipologie REV e STD. Se si analizza ad esempio il CL2, esso presenta una sensitività maggiore del 70% per entrambe le tipologie, nonostante per REV il numero totale di falsi negativi sia di 72, mentre per la tipologia STD sia di ben 483. Questo fatto è indice che i difetti in STD sono formati da un numero più elevato di candidati.
Se si osservano i singoli classificatori risulta evidente la differenza nel rapporto tra i veri positivi e le classificazioni corrette. Ad esempio, focalizzandoci su CL2 e CL3 del database STD, si noterà che il primo classifica correttamente cinque difetti in più rispetto al secondo. Se si considera però il totale dei veri positivi, il primo ne individua ben 49 in più, che rappresenta quasi il 40% del totale dei veri positivi individuati da CL3. Tale fatto porta ad una difficoltà nello studio e nella comprensione del comportamento dei classificatori, poiché non si ha necessariamente una corrispondenza lineare tra i veri positivi e le classificazioni corrette. Tale comportamento è però intrinseco nella definizione di classificazione corretta da cui non ci si può allontanare, poiché nasce dall'esigenza di voler individuare il difetto nell'immagine.
In precedenza inoltre avevamo osservato che un classificatore con un C elevato comporta un margine più piccolo e conseguentemente ad una probabile diminuzione del numero di support vector. Inoltre tra un γ molto alto, quindi molto selettivo, e uno più piccolo il primo tenderà sempre ad avere un maggior numero di support vector. Tali osservazioni risultano veritiere per i nostri classificatori. In tabella 3.8 sono riportati i numeri dei support vectors divisi per sottoclassificatori. La seconda coppia di classificatori, che presenta un γ più elevato e un C piccolo, possiede circa il doppio del numero di support vectors rispetto alla prima coppia. É interessante inoltre confrontare il numero di SV per i classificatori CL1 e CL2. Tali classificatori sono caratterizzati dallo stesso valore di C ma da un diverso valore di γ. La diversità nel numero dei SV, che si evince in tabella, fa intuire che quanto minore è la selettività del kernel RBF, tanto maggiore sarà l'energia necessaria ad ottenere un iperpiano che separi le classi, portando al minor numero di misclassificazioni. Tale ipotesi è suffragata, ad esempio, dal numero di misclassificazioni commesse nella fase di training dal sottoclassificatore dei difetti lunghi che sono sei per il CL1 (γ minore) e quattro per il CL2 (γ maggiore).

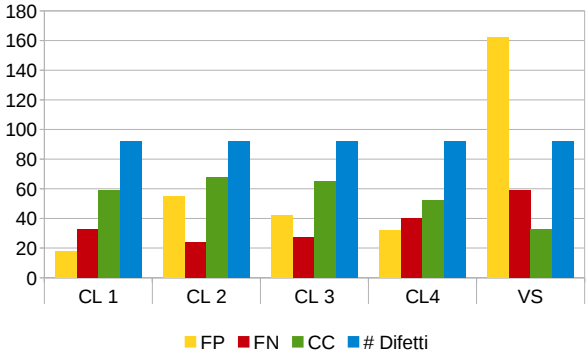


Fig. 3.3: Test 1, risultati del dataset di test REV

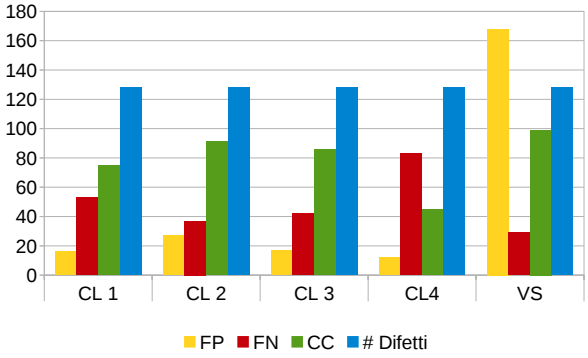


Fig. 3.4: Test 1, risultati del dataset di test STD

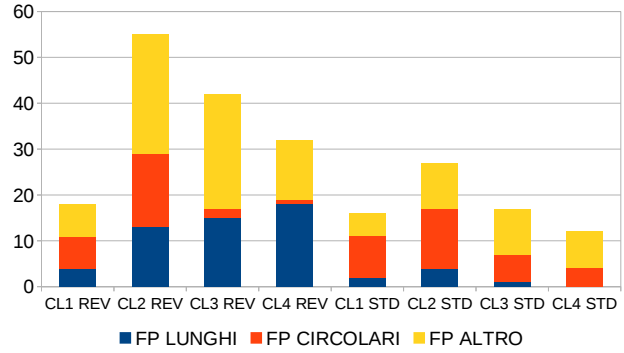


Fig. 3.5: Test 1, Composizione Falsi Positivi

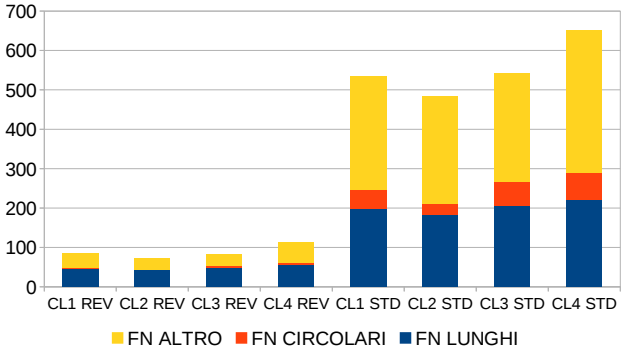


Fig. 3.6: Test 1, Composizione Falsi Negativi

Dagli istogrammi di Fig. 3.3, 3.4, 3.6 e 4.6 si può notare come gli andamenti dei valori di falsi positivi, falsi negativi e classificazioni corrette siano speculari per le due tipologie REV e STD. Per entrambi i datataset di test infatti, il classificatore che presenta il valore più alto di sensitività è CL2 che contemporaneamente risulta quello con il più alto numero di falsi positivi (Fig. 4.6). Valori molto simili si riscontrano per il classificatore CL3. Da un confronto più completo tra questi due classificatori si denota come le differenze di prestazioni siano dovute ai due sottoclassificatori di difetti di forma circolare e lunga. Se si osservano, ad esempio, i valori di tali classificatori di difetti, per la seconda tipologia (STD), i veri positivi determinati da CL2 risultano quasi il doppio di quelli determinati dal classificatori CL3.
Ricordando che il nostro obiettivo è ottenere il più alto valore di sensitività con una precisione non inferiore al 60%, si possono considerare i classificatori CL3 (per la tipologia REV) e CL2 (per la tipologia STD) i migliori classificatori ottenuti.
Dalla tabella 3.7 si può estrarre ulteriore informazione. In primo luogo si nota come i valori di sensitività ottenuti per la tipologia STD siano sempre inferiori ai corrispettivi classificatori per la tipologia REV. Tale fatto può essere dovuto ad una minor informazione ricavata dal database di training associato a tale tipologia. Questa ipotesi è avvalorata dalla disparità di candidati presenti nel database di training associabili alle due tipologie.
Dal confronto tra CL1 e CL4 si evince un fatto che era giá stato osservato nel caso di CL2 e CL3: per ottenere un valore di sensitività più elevato, conviene usare un γ poco selettivo, correggendolo poi con un C elevato. CL1 infatti, per entrambe le tipologie di test, presenta un numero di CC maggiore rispetto a CL4. Tale considerazione non vale invece per la precisione, il cui comportamento varia in base alla tipologia di dataset utilizzato nella fase di test.
Infine, osservando le percentuali dei falsi positivi dei singoli classificatori di difetti per ciascuna

forma, tabella 3.7, si evince che il classificatore di difetti circolari ha un peso molto rilevante in quasi tutti i classificatori. A tale scopo, come vedremo in seguito, nel test 4 si sono uniti i difetti circolari e altro in un unico insieme, al fine di cercare di migliorare la precisione dei classificatori.

	C.LUNGHI			**C.CIRCOLARI**			**C.ALTRO**		
	# pos	**# neg**	**# tot**	**# pos**	**# neg**	**# tot**	**# pos**	**# neg**	**# tot**
CL1	12	13	25	5	4	9	6	8	14
CL2	9	9	18	2	3	5	4	5	9
CL3	17	26	43	6	6	12	12	18	30
CL4	17	45	62	7	7	14	13	36	49

Tabella 3.8: Test1, composizione dei support vector dei quattro classificatori utilizzati (# pos corrisponde al numero dei SV appartenenti alla positive class, # neg al numero dei SV appartenenti alla negative class e # tot al numero totale dei SV)

Come ultima analisi di questo test, si è effettuata un'analisi visiva sulle immagini di outuput dei quattro classificatori. Da tale analisi si evince l'esistenza di un zoccolo duro ossia, un'insieme di difetti che tutti e quattro i classificatori riescono a classificare correttamente. Di tali insiemi fanno parte i trucioli di grandi dimensioni, molte delle deformazioni presenti sul bordo dell'oggetto nella tipologia REV, molte delle gocce d'acqua situate al centro dell'immagine per la tipologia STD e la polvere ferrosa. Al contrario ci sono delle tipologie di difetti che i classificatori in questa fase non sembrano riuscire a classificare correttamente e tra essi sono presenti i difetti chiamati ad S e le ammaccature presenti nella tipologia REV.

3.6.2 Secondo Test

Tipologia di Features: Si sono usate le stesse features del test uno, ossia:

$$fv = [MEAN_{int} \quad VAR_{int} \quad MEAN_{ext} \quad VAR_{ext}]$$

Database Training: Database di training A

Tipologia di Kernel utilizzato: Kernel RBF

Schema di classificazione: Schema 1

Per definizione, il *bounding box* è il più piccolo rettangolo che inscrive il blob del candidato. Da un'analisi visiva preliminare si è notato come in alcuni casi il background presente nel bounding box risultava formato da pochi pixel. Si è ipotizzato che tale situazione potesse inficiare il risultato della classificazione alterando le feature del background, perciò si è testato se, tramite un allargamento del bounding box di 8, 16, 32 o 64 pixel in altezza e in larghezza, si riesca ad ottenere dei miglioramenti nei risultati dei classificatori rispetto al primo test. Si fa presente che tale aumento di dimensioni non causa variazioni nella classificazione di forma, in quanto i descrittori geometrici vengono determinati sulle dimensioni del blob del candidato, che non subirà variazioni.
Di seguito verranno analizzati solamente i classificatori CL2 e CL3, cioè quelli che nel primo test risultano avere il più alto valore di sensitività.

Nelle Figure 3.7 e 3.8 sono riportate sensitività e precisione al variare di delta che indica la variazione delle dimensioni del boundingbox per il dataset di test REV. Dal grafico della sensitività si denota come l'andamento al variare di delta risulta crescente per entrambi i classificatori. Il massimo dei miglioramento si ottiene per per CL2 con delta pari a 32 e per CL4 con delta 64. Tali miglioramenti, com'è possibile vedere dalle tabelle 3.10 e 3.9, sono dovuti dall'aumento dei veri positivi per i candidati circolari e lunghi. Ci si concentrerà ora soprattutto sul miglioramento del

		Δ=0	Δ=8	Δ=16	Δ=32	Δ=64
GLOBALE	**FP**	42(0,75%)	45(0,15%)	32(0,05%)	21(0,05%)	34(0,10%)
	FN	27	29	26	22	16
	CC	65	63	66	70	76
	# Difetti	92	92	92	92	92
	Sensitività	70,65%	68,48%	71,74%	76,09%	82,61%
	Precisione	60,75%	58,33%	67,35%	76,92%	69,09%
LUNGHI	**FP**	15(2,64%)	3(0,53%)	1(0,18%)	1(0,18%)	2(0,35%)
	FN	49	47	50	43	37
	VP	42	44	41	48	54
CIRC.	**FP**	2(2,27%)	1(1,14%)	1(1,14%)	6(6,82%)	15(17,05%)
	FN	3	4	3	1	1
	VP	9	8	9	11	11
ALTRO	**FP**	25(1,87%)	41(3,07%)	30(2,25%)	14(1,05%)	17(1,27%)
	FN	30	28	26	31	31
	VP	43	45	47	42	42

Tabella 3.9: Test 2, risultati ottenuti dal classificatore CL3 con il dataset REV

		Δ=0	Δ=8	Δ=16	Δ=32	Δ=64
GLOBALE	**FP**	55(0,65%)	52(0,60%)	69(0,40%)	55(0,15%)	31(0,20%)
	FN	24	24	22	20	22
	CC	68	68	70	72	70
	# Difetti	92	92	92	92	92
	Sensitività	73,91%	73,91%	76,09%	78,26%	76,09%
	Precisione	55,28%	56,67%	50,36%	56,69%	69,31%
LUNGHI	**FP**	13(2,29%)	12(2,11%)	8(1,41%)	3(0,53%)	4(0,70%)
	FN	43	41	42	40	44
	VP	48	50	49	51	47
CIRC.	**FP**	16(18,18%)	17(19,32%)	28(31,82%)	31(35,23%)	25(28,41%)
	FN	0	0	0	0	0
	VP	12	12	12	12	12
ALTRO	**FP**	26(1,95%)	23(1,72%)	33(2,47%)	21(1,57%)	2(0,15%)
	FN	29	30	30	31	34
	VP	44	43	43	42	39

Tabella 3.10: Test 2, risultati ottenuti dal classificatore CL2 con il dataset REV

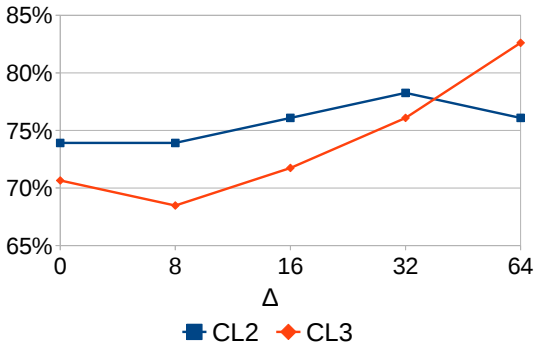


Fig. 3.7: Test 2, Andamento della sensitività al variare di Δ per il dataset di test REV

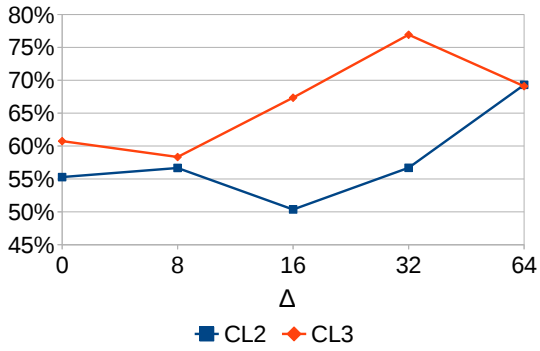


Fig. 3.8: Test 2, Andamento della precisione al variare di Δ per il dataset di test REV

sottoclassificatore di difetti lunghi. Dall'analisi delle immagini di output è emerso un fatto molto interessante. Nel primo test si è parlato di uno zoccolo duro, ossia di un certo numero di difetti che si riesce a classificare correttamente con entrambi i classificatori. Tale insieme continua a presentarsi ma, mentre l'aumento di sensitività di CL2 è causato soprattutto dall'individuazione di piccoli frammenti di metallo, l'aumento sostanziale di CL3 è dato dall'individuazione di ben 8 su 11 difetti ad S, dove nel primo test con il medesimo classificatore se ne erano individuati solo 2. Quest'ultimo fatto risulta molto interessante in quanto tali difetti si credevano molto difficili da determinare e ciò fa intuire le potenzialità del metodo SVM e le difficoltà nel determinare non solo il giusto settaggio dei parametri del classificatore, ma anche del vettore in ingresso all'algoritmo.
Con l'allargamento del bounding box si ha un miglioramento anche della precisione. Per entrambi i classificatori il numero di falsi positivi decresce per le forme lunghe ed altro mentre aumenta per la tipologia di difetti circolari.
Per la seconda tipologia oggetti, STD, i dati ottenuti nella fase di test risultano meno interessanti e opposti a quelli finora analizzati. I dati globali dei risultati sono riportati in tabella 3.11 dove si può notare come la sensitività risulti invariante a tale aumento di dimensioni, mentre si ha un peggioramento della precisione. Tale peggioramento trae origine da due fattori. In primo luogo, la precisione per tale dataset nel primo test risultava molto più elevata rispetto al database di test REV. Inoltre il peggioramento è dovuto, come nel caso precedente, al sottoclassificatore di difetti circolari. Tale peggioramento non viene bilanciato, come accade per la tipologia REV, dal miglioramento dei sottoclassificatori di difetti lunghi ed altro per i quali il numero di falsi positivi risultava già molto basso. In conclusione:

- l'aumento delle dimensioni del bounding box risulta una tecnica efficace per la tipologia REV. Con tale tecnica si riscontrano miglioramenti riguardanti sensitività e precisione dei classificatori.
- con CL2 e Δ=32 i miglioramenti sono dovuti ad un apparente aumento del numero di classificazioni corrette riduardanti difetti di forma lunga ma di dimensioni molto piccole
- con CL3 e Δ=64 si riescono a rilevare ben 8 su 11 difetti di forma ad S. Difetti che nel primo test non venivano classificati correttamente.
- per la tipologia STD non si hanno miglioramenti (e neanche peggioramenti) nell'adottare tale misura.
- l'analisi di questa tecnica ha messo in luce l'importanza di osservare le immagini di output per comprendere a fondo i risultati numerici riportati nelle tabelle.

		Δ=0	Δ=8	Δ=16	Δ=32	Δ=64
CL2	**FP**	27(3,02%)	32(3,58%)	54(6,03%)	68(7,60%)	41(4,58%)
	FN	37	39	37	39	53
	CC	91	89	91	89	75
	# Difetti	128	128	128	128	128
	Sensitività	71,09%	69,53%	71,09%	69,53%	58,59%
	Precisione	77,12%	73,55%	62,76%	56,69%	64,66%
CL3	**FP**	17(1,90%)	31(3,46%)	30(3,35%)	30(3,35%)	30(3,35%)
	FN	42	46	43	40	43
	CC	86	82	85	88	85
	# Difetti	128	128	128	128	128
	Sensitività	67,19%	64,06%	66,41%	68,75%	66,41%
	Precisione	83,50%	72,57%	73,91%	74,58%	73,91%

Tabella 3.11: Test 2, risultati ottenuti dai classificatore CL2 e CL3 con il dataset STD

3.6.3 Terzo Test

Tipologia di Features: Il vettore features utilizzato risulta:

$$ft = [MEAN_{int} \quad VAR_{int} \quad MEAN_{ext} \quad VAR_{ext}]$$

Database Training: Database di training B

Tipologia di Kernel utilizzato: Kernel RBF

Schema di classificazione: Schema 1

Nel primo test si era osservato come il classificatore avesse prestazioni leggermente inferiori per la tipologia di oggetti STD. Era stata individuata la causa nel minor numero di training example che riguardavano tale tipologia presenti nel dataset di training. Si è deciso quindi di aumentare i training examples riconducibili alla tipologia STD al fine di studiare:

- la variazione delle prestazioni dei classificatori al variare del database di training
- la sensibilità dei classificatori. Ossia come e quanto variano le prestazioni per un cambiamento non sostanzioso del database di training.

Nel nuovo database di training si sono aggiunte, rispetto a quello precedente, 4 immagini STD, tutte con presenza di difetti. In tabella 3.3 sono riportati i numeri dei due database. Il database B presenta l'8% in più di candidati e l'insieme dei candidati appartenenti alla classe difetto è aumentato del 30%.
Poiché i nuovi training example riguardano soltanto la seconda tipologia ci si aspetta un miglioramento prestazionale per i test su tale tipologia di oggetti. Si vogliono studiare inoltre, come variano i risultati per il database di test REV. A livello teorico i risultati non dovrebbero variare di molto. Nelle tabella 3.12 sono stati riportati i risultati del test sul database STD insieme ai valori ottenuti nel primo test.

		CL 1		**CL 2**		**CL 3**		**CL 4**	
		DAT A	**DAT B**	**DAT A**	**DAT B**	**DAT A**	**DAT B**	**DAT A**	**DAT B**
GLOBALE	**FP**	16(1,79%)	11(1,23%)	27(3,02%)	44(4,92%)	17(1,90%)	16(1,79%)	13(1,45%)	16(1,79%)
	FN	53	53	37	37	42	42	85	65
	CC	75	75	91	91	86	86	43	63
	# Dif.	128	128	128	128	128	128	128	128
	Sens.	58,59%	58,59%	71,09%	71,09%	67,19%	67,19%	33,59%	49,22%
	Prec.	82,42%	87,21%	77,12%	67,41%	83,50%	84,31%	76,79%	79,75%
LUNGHI	**FP**	2(0,93%)	2(0,93%)	4(1,86%)	17(7,91%)	1(0,47%)	1(0,47%)	0(0,00%)	1(0,47%)
	FN	198	197	182	176	206	175	221	214
	VP	35	36	51	57	27	58	12	19
CIRC.	**FP**	9(10,11%)	8(8,99%)	13(14,61%)	12(13,48%)	6(6,74%)	6(6,74%)	5(5,62%)	5(5,62%)
	FN	49	49	29	29	60	60	72	72
	VP	49	50	69	70	39	39	27	27
ALTRO	**FP**	5(0,85%)	1(0,17%)	10(1,69%)	15(2,54%)	10(1,69%)	9(1,52%)	8(1,35%)	10(1,69%)
	FN	287	335	272	257	275	252	362	334
	VP	108	59	123	137	119	142	32	60

Tabella 3.12: Test 3, Database di test STD

Dai risultati globali si denota come non si ottiene nessun miglioramento di sensitività per i classificatori CL2 e CL3. Se si analizzano invece i sottoclassificatori ci si accorge che l'utilizzo del nuovo database di training comporta variazioni confinate nei sottoclassificatori di difetti di forma lunga ed altro. Per entrambi i sottoclassificatori si hanno miglioramenti più consistenti per CL3 il quale non subisce variazioni sulla precisione, al contrario di CL2 in cui si registra un aumento di falsi positivi. Per tale motivo si può ritenere che l'utilizzo nel nuovo database migliori

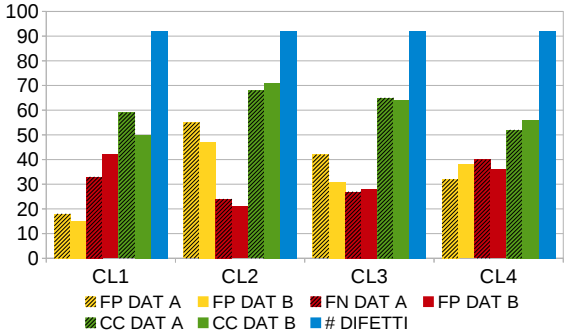


Fig. 3.9: Test 3, confronto dei risultati ottenuti sul dataset di test REV con i quattro classificatori allenati con i diversi database di training

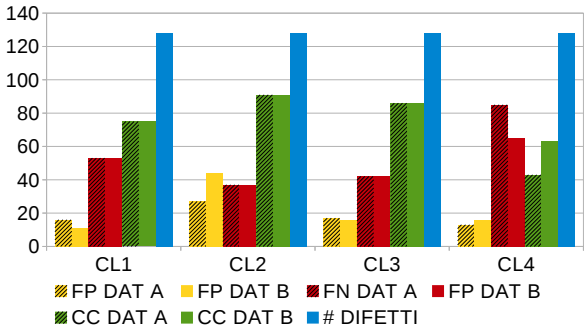


Fig. 3.10: Test 3, confronto dei risultati ottenuti sul dataset di test STD con i quattro classificatori allenati con i diversi database di training

le prestazioni per quanto riguarda CL3 e peggiori quelle del classificatore CL2. Analizzando ora gli istogrammi riguardanti il database REV, riportati in Figura 3.9 e confrontando CL2 e CL3, si vede come il nuovo dataset di training ha effetti positivi anche per tale tipologia. A discapito di una sensitività che a livello globale non varia in maniera sostanziale, si ha in entrambi i casi un netto miglioramento della precisione.

Risulta interessante studiare la variazione del numero di support vector dei nuovi classificatori, tabella 3.13. Si può notare come un aumento del database di training abbia portato anche un aumento di tali vettori, sintomo che il nuovo database contiene nuova informazione. Informazione che, com'era evidente dal risultato dei classificatori, non riguarda i difetti di forma circolare il cui numero di support vector non ha subito modifiche. Un analisi della cardinalità degli insiemi dei support vector può risultare ed è un analisi limitata, ma può risultare significativa rispetto alla domanda se è stata inserita nuova informazione. Non risponderà comunque alla domanda se tale informazione è buona o meno.
In conclusione tale test ci porta a sostenere alcune conclusioni:

- in termini di sensitività è conveniente usare classificatori con γ piccolo e C grande se il numero di training examples è piccolo. All'aumentare della cardinalità del dataset di training è conveniente usare valori di γ più selettivi;
- l'aumento del numero di training example porta ad ottenere miglioramenti nelle prestazioni dei classificatori. Nel nostro caso, però, non è detto che i singoli miglioramenti dei sottoclassificatori si traducano in miglioramenti delle prestazioni globali del classificatore.

	C.LUNGHI			**C.CIRCOLARI**			**C.ALTRO**		
	# pos	**# neg**	**# tot**	**# pos**	**# neg**	**# tot**	**# pos**	**# neg**	**# tot**
CL1	14	16	38	5	4	9	14	16	30
CL2	10	9	19	2	3	5	9	11	20
CL3	22	28	50	6	6	12	18	21	39
CL4	22	47	69	7	7	14	20	38	58

Tabella 3.13: Composizione dei support vector del database di training B

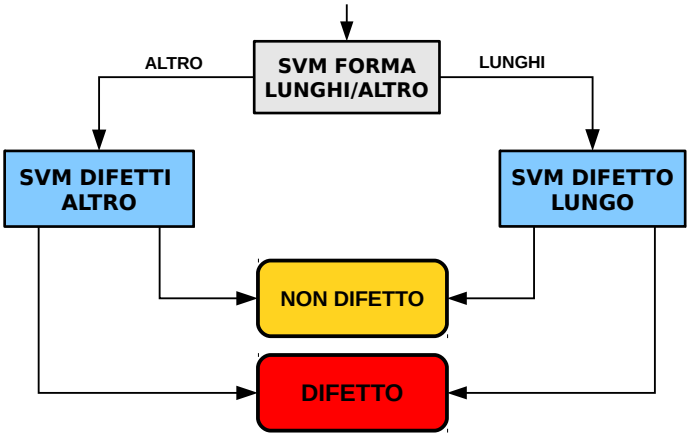


Fig. 3.11: Secondo schema di classificazione

3.6.4 Quarto Test

Tipologia di Features: Come tipologia di feaures si è utilizzato la media e la varianza del blob interno ed esterno. Il vettore features è quindi così composto:

$$featurevector = [MEAN_{int} \quad VAR_{int} \quad MEAN_{ext} \quad VAR_{ext}]$$

Database Training: Database di training A e B

Tipologia di Kernel utilizzato: Kernel RBF

Schema di classificazione: Schema 2

Nei test due e tre si è cercato di migliorare le prestazioni dei classificatori ponendosi, almeno inizialmente, l'obiettivo di migliorare la sensitività. Proviamo ora a focalizzarci sulla precisione dei classificatori. Nei grafici di Fig. 3.12 e Fig. 3.13 viene rappresentata la distribuzione dei falsi positivi in base alla forma dei candidati. Per la tipologia STD, i candidati circolari hanno un'elevata incidenza nel numero di falsi positivi, soprattutto se la si mette in relazione al numero totale di candidati circolari. Se si osservano le tabelle dei test precedenti si noterà come, la percentuale di falsi positivi circolari sia molto alta rispetto alle altre forme dei candidati. Ad esempio, in tabella 3.12, i candidati circolari hanno una media del 9% rispetto all'1,44% o all'1,63%,rispettivamente dei difetti lunghi e altro. Si può attribuire la causa di tale risultato, al basso numero di candidati circolari usati per il training. Per tale motivo si è voluto testare la variazione dei risultati dei classificatori nel caso si adotti un secondo schema di classificazione, riportato in Figura 3.11. Tale classificatore è stato strutturato in modo da suddividere i candidati in due sole classi di forma:

- $lunghi'' = lunghi'$
- $altro'' = circolari' \cup altro'$

dove con apice singolo sono indicati gli insiemi dei candidati ottenuti dal primo schema di classificazione, con doppio apice quelli ottenuti dal secondo schema.

Verranno di seguito analizzati solo i dati dei classificatori allenati con il dataset di training A in quanto i risultati hanno lo stesso andamento dei classificatori allenati con il dataset B. Tali dati sono riportati nelle tabelle 3.14 e 3.15. Utilizzando il secondo schema di classificazione si pensava che il numero di falsi positivi globali potessero diminuire anche a discapito della sensitività. Dai risultati si denota come la precisione non migliora, anzi, peggiora nella gran

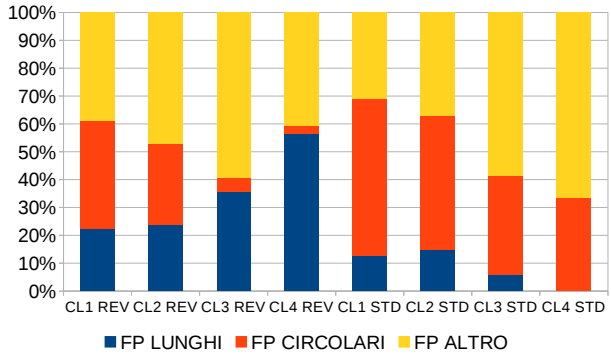


Fig. 3.12: Test 4, istogramma composizione in percentuale dei falsi positivi dei classificatori allenati con il database di training A

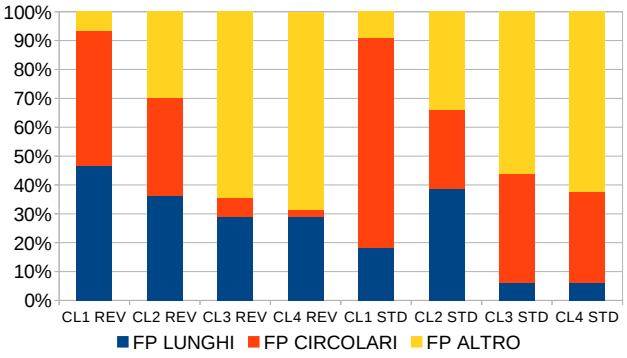


Fig. 3.13: Test 4, istogramma composizione in percentuale dei falsi positivi dei classificatori allenati con il database di training B

parte dei casi. Peggiora anche la sensitività dei classificatori, soprattutto per la la tipologia STD. Questo risultato rafforza la bontà dell'idea iniziale di dividere i candidati secondo tre tipologie di forme. Inoltre sembrerebbe avvalorare la tesi che le forme caratterizzano le features dei colori.

		CL 1		CL 2		CL 3		CL 4	
		SCH 1	**SCH 2**	**SCH 1**	**SCH 2**	**SCH 1**	**SCH 2**	**SCH 1**	**SCH 2**
GLOBALE	**FP**	18(0,90%)	25(1,26%)	55(2,76%)	53(2,66%)	42(2,11%)	42(2,11%)	32(1,61%)	67(3,37%)
	FN	33	28	24	25	27	26	40	37
	CC	59	64	68	67	65	66	52	55
	# Dif.	92	92	92	92	92	92	92	92
	Sens.	64,13%	69,57%	73,91%	72,83%	70,65%	71,74%	56,52%	59,78%
	Prec.	76,62%	71,91%	55,28%	55,83%	60,75%	61,11%	61,90%	45,08%
LUNGHI	**FP**	4(0,70%)	4(0,70%)	13(2,29%)	13(2,29%)	15(2,64%)	15(2,64%)	18(3,17%)	18(3,17%)
	FN	47	47	43	43	49	49	55	55
	VP	44	44	48	48	42	42	36	36
CIRC.	**FP**	7(7,95%)	0(0,00%)	16(18,18%)	0(0,00%)	2(2,27%)	0(0,00%)	1(1,14%)	0(0,00%)
	FN	1	0	0	0	3	0	6	0
	VP	11	0	12	0	9	0	6	0
ALTRO	**FP**	7(0,52%)	21(1,57%)	26(1,95%)	40(3,00%)	25(1,87%)	27(2,02%)	13(0,97%)	49(3,67%)
	FN	36	34	29	30	30	31	51	48
	VP	37	51	44	55	43	54	22	37

Tabella 3.14: Test 4, REV

		CL 1		CL 2		CL 3		CL 4	
		SCH 1	**SCH 2**	**SCH 1**	**SCH 2**	**SCH 1**	**SCH 2**	**SCH 1**	**SCH 2**
GLOBALE	**FP**	16(1,79%)	25(2,79%)	27(3,02%)	32(3,58%)	17(1,90%)	17(1,90%)	12(1,34%)	24(2,68%)
	FN	53	72	37	43	42	55	83	77
	CC	75	56	91	85	86	73	45	51
	# Dif.	128	128	128	128	128	128	128	128
	Sensi.	58,59%	43,75%	71,09%	66,41%	67,19%	57,03%	35,16%	39,84%
	Prec.	82,42%	69,14%	77,12%	72,65%	83,50%	81,11%	78,95%	68,00%
LUNGHI	**FP**	2(0,93%)	2(0,93%)	4(1,86%)	4(1,86%)	1(0,47%)	1(0,47%)	0(0,00%)	0(0,00%)
	FN	198	198	182	182	206	206	221	221
	VP	35	35	51	51	27	27	12	12
CIRC.	**FP**	9(10,11%)	0(0,00%)	13(14,61%)	0(0,00%)	6(6,74%)	0(0,00%)	4(4,49%)	0(0,00%)
	FN	49	0	29	0	60	0	69	0
	VP	49	0	69	0	39	0	30	0
ALTRO	**FP**	5(0,85%)	23(3,89%)	10(1,69%)	28(4,74%)	10(1,69%)	16(2,71%)	8(1,35%)	24(4,06%)
	FN	287	359	272	305	275	325	362	412
	VP	108	134	123	188	119	168	32	81

Tabella 3.15: Test 4, STD

3.6.5 Riepilogo risultati

Al fine di proseguire in maniera efficace la discussione si vogliono riassumere i risultati fin qui ottenuti. Per farlo si andranno a determinare i miglior classificatori, dove per migliore si intende il classificatore con il miglior valore di sensitività. Le strutture del classificatore, Fig. 3.2 e Fig. 3.11, permettono di andare ad interscambiare i sotto-classificatori di difetti delle varie forme al fine di determinare quello con le migliori prestazioni. Nelle tabelle 3.16 e 3.17 sono riportati i risultati e le composizioni dei migliori classificatori ottenuti. Nei test successivi si cercherà di migliorare tali prestazioni andando a variare la tipologia del vettore di feature in ingresso ai classificatori e/o si andranno ad utilizzare altre tipologie di kernel. I risultati ottenuti saranno confrontati con i migliori classificatori fin qui ricavati.

		REV	**STD**
GLOBALE	**FP**	34(1,71%)	45(5,03%)
	FN	16	30
	CC	76	98
	# Difetti	92	128
	Sensitività	82,61%	76,56%
	Precisione	69,09%	68,53%
LUNGHI	**FP**	2(0,35%)	4(1,86%)
	FN	37	156
	VP	54	77
CIRC.	**FP**	15(17,05%)	13(14,61%)
	FN	1	30
	VP	11	69
ALTRO	**FP**	17(1,27%)	28(4,74%)
	FN	31	191
	VP	42	203

Tabella 3.16: Risultati dei migliori classificatori ottenuti tramite features statistiche e kernel RBF

	REV	**STD**
LUNGHI	CL3 DELTA 64 DAT A	CL3 DELTA 0 DAT B
CIRC.	CL3 DELTA 64 DAT A	CL2 DELTA 0 DAT B
ALTRO	CL3 DELTA 64 DAT A	CL3 DELTA 0 DAT B

Tabella 3.17: Composizione dei migliori classificatori ottenuti tramite features statistiche e kernel RBF

3.6.6 Quinto Test

Tipologia di Features: Come tipologia di feaures si sono utilizzati gli istogrammi normalizzati con un diverso numero di bins pari a 20, 40 e 80:

$$H_{norm} = \left[\frac{H_1}{\sum_{i=1}^{n} H_i}, \frac{H_2}{\sum_{i=1}^{n} H_i}, \ldots, \frac{H_n}{\sum_{i=1}^{n} H_i} \right]$$

Database Training: Database di training A e B

Tipologia di Kernel utilizzato: Kernel RBF

Schema di classificazione: Schema 1 e 2

In questo test, come vettore di fetures si è utilizzato l'istogramma dell'immagine normalizzata in scala di grigio con numero di bins scelto a priori tra 20, 40 e 80. A livello di controllo si è aggiunto un terzo parametro poiché, rispetto ai parametri C e γ del classificatore SVM con Kernel RBF, si è aggiunto il numero di bins dell'istogramma. Si è deciso, sfruttando l'esperienza dei test precedenti, di scegliere un valore di γ pari a 0.1, un valore che garantisce un buon livello di selettività paragonabile al valore di CL2 dei test precedenti. Bisogna considerare inoltre che, con l'aumentare della dimensione del vettore di features, valori di γ maggiori di uno risultano troppo selettivi, portando ad ottimi risultati nella fase di training ma ottenendo nella fase di test valori di sensitività molto bassi. Si terrà costante il valore di γ, pari ad 0.1, variando il parametro C e il numero di bins. Tale test è stato effettuato utilizzando entrambi i database di training.

In Figura 3.14 e 3.15 sono riportati i grafici dei test effettuati sul database di test REV. In tali grafici e nei successivi le linee di congiunzione non indicano un andamento della sensitività, ma vengono riportate per evidenziare le variazioni della precisione e sensitività al variare del valore di C scelto. In tali grafici si è riportato con linee di colore verde i risultati del classificatore di tabella 3.16 e in colore rosso il miglior classificatore determinato in questo test. E' evidente come in termini di sensitività non si riescano a migliorare le prestazioni precedentemente riscontrate. Il classificatore con valore di sensitività più alto, risulta il classificatore con numero di bins 80, allenato con il database B e con paramenti C e γ pari ad 2^{15} e 0.1. Tale classificatore ottiene una sensitività del 74% con una precisione molto bassa, 55%. Se si osservano i grafici che fanno riferimento al database REV, si evince che:

- si ottengono valori di precisione molto bassi, solo in tre casi su nove si hanno valori al di sopra del 60% , Fig. 3.15;
- si ha un netto peggioramento delle prestazioni se si utilizza il secondo database di training soprattutto se si fa riferimento alla precisione, al contrario del test effettuato utilizzando features statistiche;
- le migliori prestazioni si ottengono con un istogramma a 20 bins;
- effettuando un confronto visivo sulle immagini dei classificatori con vettori di features statistiche o istogrammi, si evincono alcune caratteristiche: in primo luogo gli istogrammi hanno meno difficoltà a classificare correttamente alcuni difetti doppi, ossia candidati in cui il bounding box andava a considerare parti di regione di altri candidati. In tale situazione le faetures statistiche di media e varianza del background posso risultare falsate. Con gli istogrammi non si riescono ad individuare nessun difetto relativo alle ammaccature, al contrario delle features statistiche.

Nelle figure 3.16 e 3.17 sono riportati invece i grafici riguardanti i test sul database STD, per i quali:

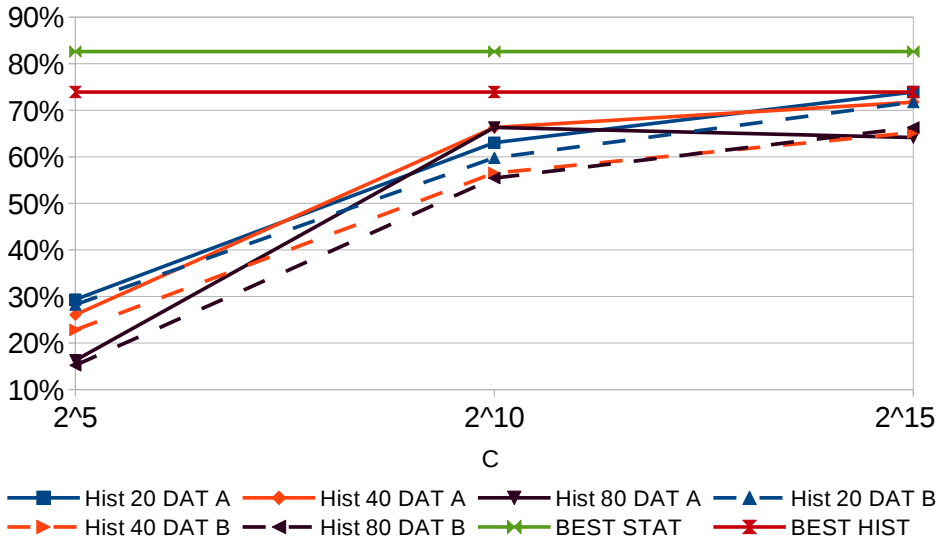


Fig. 3.14: Test 5, Andamento Sensitività Dataset di Test REV

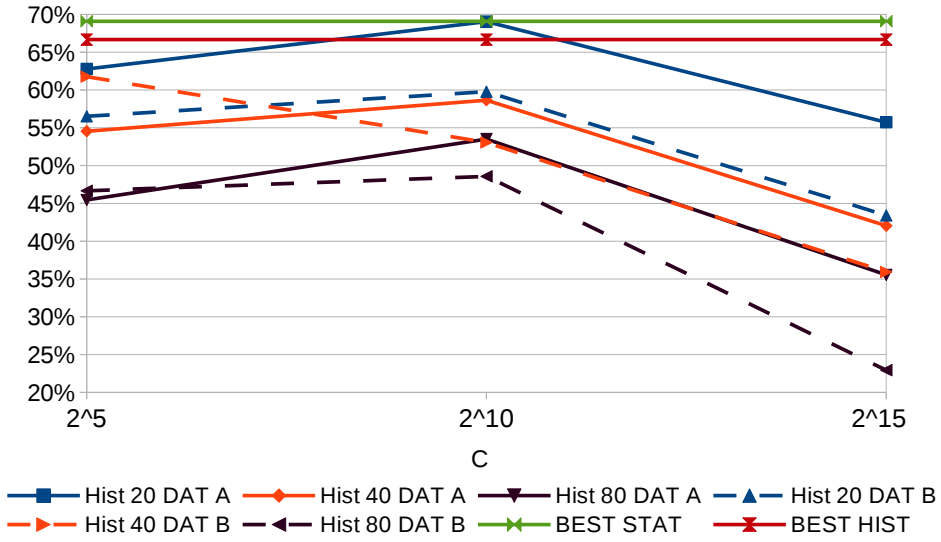


Fig. 3.15: Test 5, Andamento della Precisione del Dataset di Test REV

- si ha un netto miglioramento prestazionale riguardante la sensitività se si utilizza il secondo database di training rispetto al primo;
- la precisione dei classificatori allenati con il dataset B peggiora rispetto ai classificatori allenati con il dataset A solo per valori di C elevati;
- nuovamente il miglior valore di precisione lo si ottiene con un istogramma a 20 bins il quale però, ha bassi valori di sensitività;
- il miglior classificatore lo si ottiene con un istogramma con valori di bins pari a 40, classificatore con prestazioni inferiore rispetto al miglior classificatore ottenuto con features statistiche;
- dal controllo visivo si evince che c'è molta differenza tra i veri positivi dei candidati classificati correttamente tra le due tipologie di features. Ciò porta ad ottenere lo stesso parametro di sensitività ma attraverso classificazioni corrette di diversi difetti.

Nelle tabelle 3.18 e 3.19 sono riportate le statistiche e la composizione dei migliori classificatori riportati nei grafici precedenti. In esse si può notare, andando a confrontare singolarmente i valori

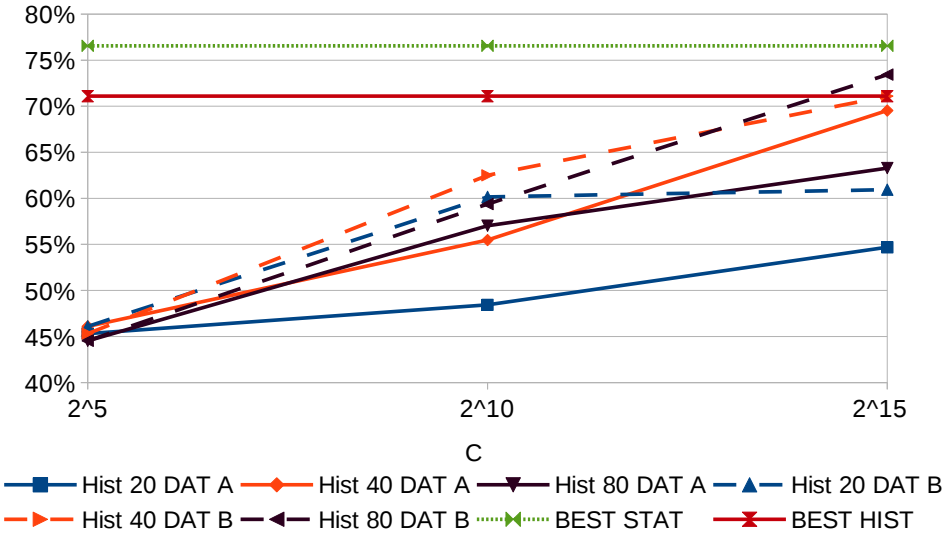


Fig. 3.16: Test 5, Andamento Sensitività Dataset di Test STD

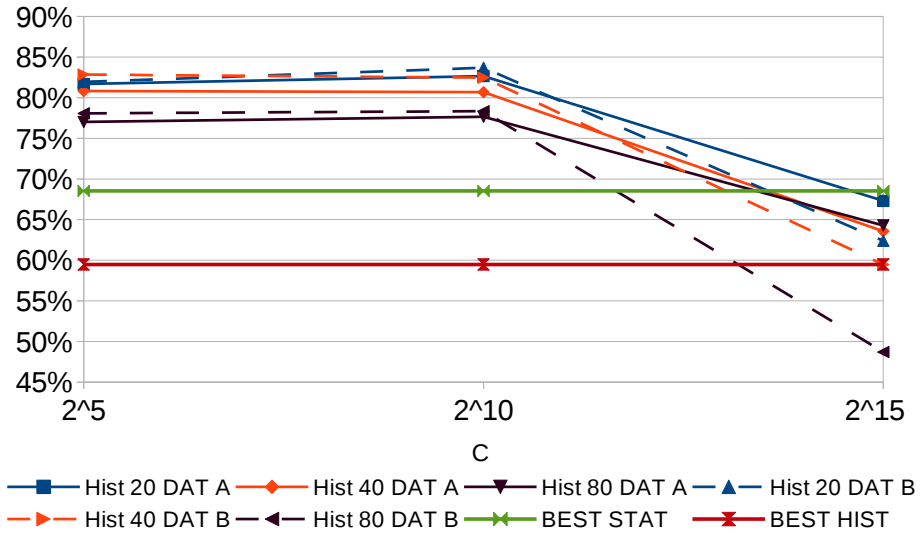


Fig. 3.17: Test 5, Andamento Precisione Dataset di Test STD

dei veri positivi dei sottoclassificatori, che l'utilizzo della seconda tipologia di features porta ad ottenete risultati peggiori. I falsi positivi si comportano in maniera diversa per le due tipologie di immagini. Mentre per le features statistiche si ha per entrambi i dataset di test che il classificatore più virtuoso risulta essere il sottoclassificatore dei difetti lunghi seguito da quello dei difetti altro e circolare, tale caratteristica non si riscontra per la seconda tipologia di features. Tale situazione può risultare sconveniente per andare a capire dove agire per migliorare le prestazioni.

Ci si vuole soffermare ulteriormente sui risultati ottenuti in questo quinto test. I classificatori ottenuti in questo test, presentano un numero di misclassificazioni totali maggiore rispetto ai classificatori ottenuti con features statistiche. Ciò è spiegabile, almeno in parte, dal comportamento della metrica (d_{RBF}) con cui il kernel RBF misura la diversità dei vettori, (3.3).

$$K_{RBF}(H, M) = \exp\left(-\gamma d_{RBF}(H, M)\right) = \exp\left(-\gamma \sum_{l=1}^{n} \|h_{c_l} - m_{c_l}\|^2\right) \tag{3.3}$$

Consideriamo gli istogrammi H1,H2 e H3 di Figura 3.18. Ad esempio, l'istogramma H2 può essere ricavato dalla stessa immagine di H1 dove, a causa di un cambio di luminosità, i colori

hanno subito uno slittamento sul bin di destra. Situazione che possiamo ritenere non del tutto estranea alla nostra applicazione. É evidente come i primi due istogrammi siano più simili tra loro rispetto al terzo preso in considerazione. Se però, si calcola il valore della metrica associabile al kernel RBF si ottiene:

$$d_{RBF}(H_1, H_2) = 6P^2 > d_{RBF}(H_1, H_3) = d_{RBF}(H_2, H_3) = 4P^2$$

che risulta l'opposto di quello che ci si aspetta. Tale metrica inoltre tende a sovrastimare la similarità nel caso di una distribuzione piatta, ossia una distribuzione che possiede molti bin con valori piccoli come nel caso dell'istogramma I3 di Figura 3.18. Anche in questo caso, se andiamo a determinare la metrica, si ottiene:

$$d_{RBF}(I_1, I_2) = 1.88 > d_{RBF}(I_1, I_3) = 1.58 > d_{RBF}(I_2, I_3) = 1.2$$

che ancora una volta risulta l'opposto di quello che ci si aspetterebbe. Tali esempi vanno a spiegare il fatto che, in termini di precisione, i risultati migliori li si ottengano con istogrammi con i valori di bins pari a 20, in quanto riducendo il numero di bins si va a rendere questa tipologia di features più solida rispetto ad esempio ad un cambio di luminosità. Tali esempi possono spiegare il peggioramento dei risultati dei classificatori ottenuti con questa tipologia di features rispetto a quella statistica. Con le features statistiche tali errori non si commetterebbero. Ad esempio se si considera la prima terna di istogrammi si otterrebbero le seguenti features:

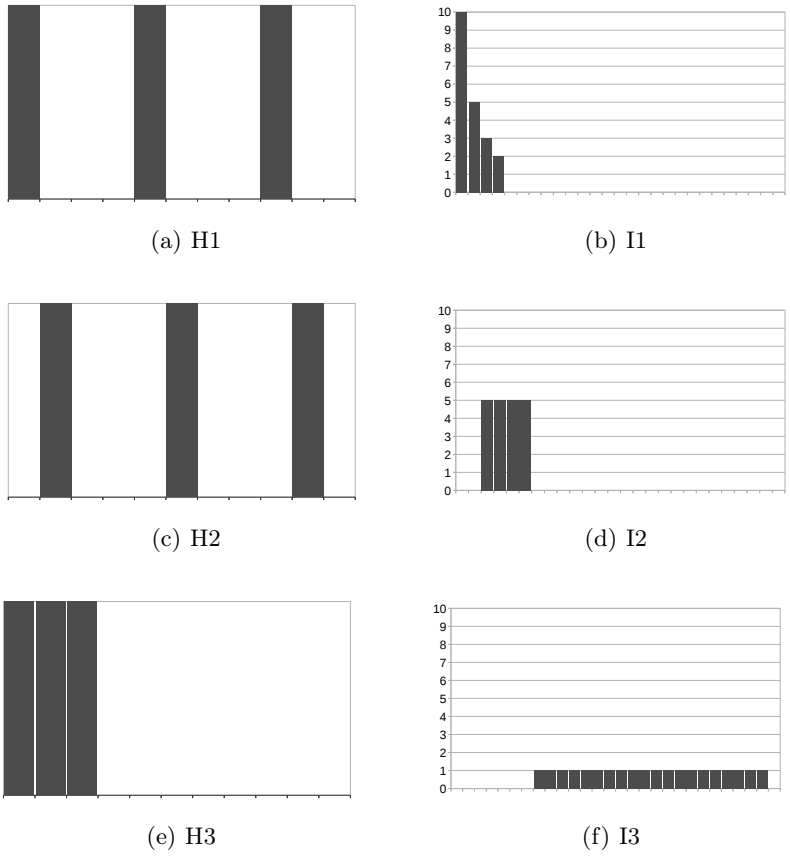
- $$H_1 = \begin{bmatrix} 0.5P & 0.16P^2 \end{bmatrix}$$
- $$H_1 = \begin{bmatrix} 0.6P & 0.16P^2 \end{bmatrix}$$
- $$H_1 = \begin{bmatrix} 0.2P & 0.01P^2 \end{bmatrix}$$

che porterebbero ad ottenere:

- $$d_{RBF}(H_2, H_3) = \gamma(0.16P^2 + 0.0225P^4)$$
- $$d_{RBF}(H_1, H_3) = \gamma(0.09P^2 + 0.0225P^4)^2$$
- $$d_{RBF}(H_1, H_2) = \gamma(0.01P^2)$$

Si ottiene una stima delle distanze corretta:

$$d_{RBF}(H_2, H_3) > d_{RBF}(H_1, H_3 > d_{RBF}(H_1, H_2)$$



(a) H1 (b) I1

(c) H2 (d) I2

(e) H3 (f) I3

Fig. 3.18: Test 6, istogrammi di esempio

		REV		STD	
		ISTOG.	**STAT.**	**ISTOG.**	**STAT.**
GLOBALE	**FP**	34(1,71%)	34(1,71%)	62(6,93%)	45(5,03%)
	FN	24	16	37	30
	CC	68	76	91	98
	# Dif.	92	92	128	128
	Sens.	73,91%	82,61%	71,09%	76,56%
	Prec.	66,67%	69,09%	59,48%	68,53%
LUNGHI	**FP**	15(2,64%)	2(0,35%)	7(3,26%)	4(1,86%)
	FN	41	37	181	156
	VP	50	54	52	77
CIRC.	**FP**	10(11,36%)	15(17,05%)	22(24,72%)	13(14,61%)
	FN	1	1	28	30
	VP	11	11	71	69
ALTRO	**FP**	9(0,67%)	17(1,27%)	33(5,58%)	28(4,74%)
	FN	34	31	268	191
	VP	39	42	126	203

Tabella 3.18: Risultati dei migliori classificatori ottenuti tramite features di istogrammi e kernel RBF

	REV	**STD**
LUNGHI	DAT A HIST 20 $\gamma = 0.1$ C $=2^{15}$	DAT B HIST 40 $\gamma = 0.1$ C $= 2^{15}$
CIRC.	DAT A HIST 40 $\gamma = 0.1$ C $=2^{10}$	DAT B HIST 40 $\gamma = 0.1$ C $= 2^{15}$
ALTRO	DAT A HIST 20 $\gamma = 0.1$ C $=2^{15}$	DAT B HIST 40 $\gamma = 0.1$ C $= 2^{15}$

Tabella 3.19: Composizione dei migliori classificatori ottenuti tramite features di istogrammi e kernel RBF

3.6.7 Sesto Test

Tipologia di Features: Come tipologia di feaures si utilizzeranno gli istogrammi con numero di bin pari a 20,40 e 80

Database Training: Database di training A e B

Tipologia di Kernel utilizzato: Kernel χ^2, laplaciano e HIK

Schema di classificazione: Schema 1 e 2

Nel test precedente si è visto che, utilizzando come feature vector l'istogramma del colore, la metrica utilizzata dal kernel RBF porta in alcuni casi a sbagliare a misurare la similarità tra i vettori. In questo test si sono voluti testare tre diversi kernel con la stessa tipologia di feature vector del precedente test. I primi due kernel, χ^2 e laplaciano, sono stati scelti poiché dovrebbero garantire prestazioni migliori rispetto al kernel RBF in quanto sono meno soggetti a casi come quelli precedenti. Questi due kernel, come RBF, sono kernel esponenziali e variano tra loro in base alla metrica con cui vanno a misurare la somiglianza tra i vettori. Tale metriche corrispondono a:

$$K_{Laplacian}(H, M) = \exp\left(-\gamma d_{L1}(H, M)\right) = \exp\left(-\gamma \sum_{l=1}^{n} |h_{c_l} - m_{c_l}|\right) \tag{3.4}$$

$$K_{\chi^2}(H, M) = \exp\left(-\gamma d_{\chi^2}(H, L)\right) = \exp\left(-\gamma \sum_{l=1}^{n} \frac{(h_{c_l} - m_{c_l})^2}{h_{c_l} + m_{c_l}}\right) \tag{3.5}$$

Proviamo ora a testare i due nuovi Kernel per le situazioni riportate in Figura 3.18. Se consideriamo la prima terna di istogrammi H_1, H_2 e H_3 si ottiene:

$$d_{RBF}(H_1, H_2) = 6P^2 > d_{RBF}(H_1, H_3) = d_{RBF}(H_2, H_3) = 4P^2$$
$$d_{L_1}(H_1, H_2) = 6P > d_{L_1}(H_1, H_3) = d_{L_1}(H_2, H_3) = 4P$$
$$d_{\chi^2}(H_1, H_2) = 6P > d_{\chi^2}(H_1, H_3) = d_{\chi^2}(H_2, H_3) = 4P$$

dove, come nel caso del kernel precedente, si sbaglia a stimare la similarità tra gli istogrammi, mentre però, per il kernel RBF si ha un incremento quadratico, per i due nuovi kernel l'incremento risulta lineare. Si commette quindi un errore più piccolo. Se si considera invece la seconda terna di istogrammi I_1, I_2 e I_3 si ha:

$$d_{RBF}(I_1, I_2) = 1.88 > d_{RBF}(I_1, I_3) = 1.58 > d_{RBF}(I_2, I_3) = 1.2$$
$$d_{L_1}(I_1, I_3) = d_{L_1}(I_2, I_3) = 4 > d_{L_1}(I_1, I_2) = 3$$
$$d_{\chi^2}(I_1, I_3) = d_{\chi^2}(I_2, I_3) = 4 > d_{\chi^2}(I_1, I_2) = 2.7$$

In tal caso i due kernel stimano in maniera esatta la somiglianza tra i diversi istogrammi al contrario della metrica di RBF. Ci si aspetterà da tali kernel delle prestazioni più elevate per quanto riguarda la precisione.
Insieme a questi due kernel si è voluto testare anche il kernel HIK. Ricordiamo inoltre che non esiste un modo per andare a misurare a priori la bontà di un kernel rispetto al vettore delle feature utilizzato. Nelle tabelle 3.22 e 3.23 si sono riassunti alcuni dei risultati ottenuti in questo test che mettono in risalto i comportamenti dei diversi Kernel. Come visto nel test precedente, i migliori risultati, per la tipologia REV, si ottengono allenando i classificatori con con il dataset di training A, per la tipologia STD, invece, allenando i classificatori con il dataset B. I parametri usati per allenare i classificatori i cui risultati sono stati riportati in tabella 3.22 sono:

- Kernel RBF: $C = 2^{10}$ e $\gamma = 0.1$
- Kernel χ^2: $C = 2^7$ e $\gamma = 0.1$
- Kernel L1: $C = 2^5$ e $\gamma = 1$
- Kernel HIK: $C = 2^5$

Tali parametri risultano i medesimi anche per i classificatori di tabella 3.23 tranne che per il kernel χ^2 per il quale i parametri risultano $C = 2^7$ e $\gamma = 0.5$. I valori dei parametri dei classificatori sono stati scelti per garantire, ove possibile, un valore della sensitività del 60%. Tale scelta dei parametri mette in evidenza un primo aspetto delle ipotesi riportate ad inizio test. Infatti, al fine di ottenere un parametro di sensitività del 60%, il kernel RBF ha bisogno di un valore di C molto più alto rispetto agli altri kernel, indice di una difficoltà del kernel a classificare correttamente le features. Tale aspetto inoltre si ripercuote sul range dei valori di C utilizzabili per gli altri kernel. Infatti i valori di C utilizzati per i 3 nuovi kernel, rappresentano un valore limite oltre al quale, aumentando il valore di C, i risultati nella fase di test subiscono piccole variazioni. Al contrario, per il kernel RBF all'aumentare di C, i risultati subiscono notevoli cambiamenti come si può notare dai risultati del test precedente. Dalle due tabelle si possono estrarre alcune considerazioni generali:

- il kernel χ^2 risulta quello con sensitività più elevata nel maggior parte dei casi;
- il kernel L_1 risulta quello con precisione più alta nella maggior parte dei casi;
- per tre nuovi kernel la sensitività cresce all'aumentare del numero di bin, e la precisione rimane costante e sempre al di sopra del 60%, al contrario di quello che accadeva per il kernel RBF ;
- il miglioramento della sensitività risulta dovuto al sottoclassificatore di difetti altro. In tabella 3.23 si può notare come tale sottoclassificatore per il kenrnel RBF tende a classificare correttamente almeno 25% in meno dei candidati rispetto agli altri sottoclassificatori. Questo comportamento, con diverse percentuali, si ha anche nel caso della tipologia REV;
- il migliore classificatore risulta quello con kernel χ^2 per il quale si ottengono i più alti valori di sensitività, con valori di precisione costantemente sopra al 60%.
- a livello di sottoclassificatori ancora una volta quelli che ottengono il maggior numero di veri positivi risultano essere per entrambi i dataset di test i classificatori χ^2 e HIK. É da sottolineare che quest'ultimo classificatore risulta insieme ad L_1 uno dei classificatori con il più basso numero di falsi positivi.

Come nei test precedenti, si è cercato di combinare i sottoclassificatori al fine di migliorare le prestazioni dei diversi kernel e i risultati sono riportati nelle tabelle 3.24 e 3.25. Tali tabelle sembrano controvertire le osservazioni fatte in precedenza. Infatti per entrambi i dataset di test i risultati migliori, in termini di sensitività, si ottengono dal classificatore RBF. É da sottolineare però, che tali classificatori sono frutto di una fase di tuning molto elaborata e dispendiosa in quanto frutto di tre step. Per prima cosa si sono andati a svolgere una fase di test per determinare il range di parametri migliori. Una volta determinati tali parametri, si sono raccolti i risultati ed infine tali risultati sono stati incrociati per determinare i valori riportati in tabella. Ciò implica che il kernel RBF riesce ad estrarre più informazione dalle features ma, se non si ha il tempo o la possibilità di una lunga fase di tuning, il kernel χ^2 e HIK hanno prestazioni migliori; inoltre o presentano un range minore di valori dei parametri da impostare, kernel χ^2, o hanno un minor numero di parametri da impostare, kernel HIK. Caratteristiche che in un approccio iniziale ai problemi possono fare la differenza. C'è un ultimo aspetto non banale da considerare. Mentre

	χ^2	L_1	**HIK**
LUNGHI	# Bins = 20 γ=0,1 C=2^7	# Bins = 80 γ=1 C=2^5	# Bins = 40 C=2^5
CIRC.	# Bins = 20 γ=0,1 C=2^7	# Bins = 80 γ=1 C=2^5	# Bins = 20 C=2^5
ALTRO	# Bins = 80 γ=0,1 C=2^7	# Bins = 80 γ=1 C=2^5	# Bins = 80 C=2^5

Tabella 3.20: Composizione dei migliori classificatori ottenuti tramite features di istogrammi per il database STD

	χ^2	L_1	**HIK**

LUNGHI	# Bins=80
	C=2^7
	γ= 0.5
		# Bins = 80
		C=2^5
		γ= 0.5
			# Bins = 80
			C=2^5

CIRC.	# Bins = 80
	C=2^7
	γ= 0.5
		# Bins = 40
		C=2^5
		γ= 0.5
			# Bins = 80
			C=2^5

ALTRO	# Bins = 80
	C=2^7
	γ= 0.5
		# Bins = 20
		C=2^5
		γ= 0.5
			# Bins = 80
			C=2^5

Tabella 3.21: Composizione dei migliori classificatori ottenuti tramite features di istogrammi per il database STD

con RBF c'è il rischio, come si può vedere dai dati del test cinque, che sbagliando il parametro C si ottengano discreti valori di sensitività ma pessimi valori di precisione, i tre kernel provati in questo test non hanno mai evidenziato tale problema. Anche testando i tre nuovi kernel con valori di C elevati, pari a quelli provati per il kernel RBF, la precisione è sempre rimasta al di sopra del 60%. Ciò rende questi classificatori più affidabili.

		HIST 20				HIST40				HIST 80			
		RBF	χ^2	L_1	**HIK**	**RBF**	χ^2	L_1	**HIK**	**RBF**	χ^2	L_1	**HIK**
GLOBALE	FP	26(1,31%)	32(1,61%)	21(1,06%)	28(1,41%)	43(2,16%)	38(1,91%)	22(1,11%)	30(1,51%)	53(2,66%)	41(2,06%)	28(1,41%)	34(1,71%)
	FN	34	29	38	35	31	28	31	33	31	29	28	28
	CC	58	63	54	57	61	64	61	59	61	63	64	64
	# Dif.	92	92	92	92	92	92	92	92	92	92	92	92
	Sens.	63,04%	68,48%	58,70%	61,96%	66,30%	69,57%	66,30%	64,13%	66,30%	68,48%	69,57%	69,57%
	Prec.	69,05%	66,32%	72,00%	67,06%	58,65%	62,75%	73,49%	66,29%	53,51%	60,58%	69,57%	65,31%
LUNGHI	FP	13(2,29%)	10(1,76%)	6(1,06%)	8(1,41%)	20(3,52%)	18(3,17%)	12(2,11%)	15(2,64%)	20(3,52)	15(2,64%)	16(2,82%)	17(2,99%)
	FN	51	48	54	53	52	48	49	47	51	51	49	49
	VP	40	43	37	38	39	43	42	44	40	40	42	42
CIRC.	FP	8(9,09%)	12(13,64%)	3(3,41%)	4(4,55%)	10(11,36%)	12(13,64%)	7(7,95%)	8(9,09%)	12(13,64%)	12(13,64%)	7(7,95%)	8(9,09%)
	FN	2	1	2	1	1	1	1	2	2	2	2	2
	VP	10	11	10	11	11	11	11	10	10	10	10	10
ALTRO	FP	5(0,37%)	10(0,75%)	12(0,90%)	16(1,20%)	13(0,97%)	8(0,60%)	3(0,22%)	7(0,52%)	21(1,57%)	14(1,05%)	5(0,37%)	9(0,67%)
	FN	42	33	40	38	34	31	35	35	36	30	32	30
	VP	31	40	33	35	39	42	38	38	37	43	41	43

Tabella 3.22: Test 6, Risultati del dataset di test REV allenati con il database di training A

		HIST 20				HIST40				HIST 80			
		RBF	χ^2	L_1	**HIK**	**RBF**	χ^2	L_1	**HIK**	**RBF**	χ^2	L_1	**HIK**
GLOBALE	FP	15(1,68%)	33(3,69%)	23(2,57%)	31(3,46%)	17(1,90%)	29(3,24%)	18(2,01%)	22(2,46%)	21(2,35%)	40(4,47%)	19(2,12%)	25(2,79%)
	FN	51	50	49	48	48	43	47	52	52	41	46	43
	CC	77	78	79	80	80	85	81	76	76	87	82	85
	# Dif.	128	128	128	128	128	128	128	128	128	128	128	128
	Sens.	60,16%	60,94%	61,72%	62,50%	62,50%	66,41%	63,28%	59,38%	59,38%	67,97%	64,06	%66,41%
	Prec.	83,70%	70,27%	77,45%	72,07%	82,47%	74,56%	81,82%	77,55%	78,35%	68,50%	81,19%	77,27%
LUNGHI	FP	2(0,93%)	4(1,86%)	3(1,40%)	3(1,40%)	2(0,93%)	7(3,26%)	2(0,93%)	5(2,33%)	4(1,86%)	9(4,19%)	2(0,93%)	2(0,93%)
	FN	189	176	191	199	196	186	200	203	199	176	192	193
	VP	44	57	42	34	37	47	33	30	34	57	41	40
CIRC.	FP	9(10,11%)	8(8,99%)	7(7,87%)	8(8,99%)	10(11,24%)	10(11,24%)	9(10,11%)	8(8,99%)	12(13,48%)	11(12,36%)	11(12,36%)	11(12,36%)
	FN	45	49	45	46	43	42	39	40	44	41	41	39
	VP	54	50	54	53	56	57	60	59	55	58	58	60
ALTRO	FP	4(0,68%)	21(3,55%)	13(2,20%)	20(3,38%)	5(0,85%)	12(2,03%)	7(1,18%)	9(1,52%)	5(0,85%)	20(3,38%)	6(1,02%)	12(2,03%)
	FN	307	283	279	267	313	262	291	287	325	268	291	270
	VP	87	111	115	127	81	132	103	107	69	126	103	124

Tabella 3.23: Test 6, Risultati del dataset di test STD allenati con il database di training B

TIP.FEAT.		ISTOGRAMMI				STAT.
KERNEL		**RBF**	χ^2	L_1	**HIK**	**REF.**
GLOBALE	**FP**	34(1,71%)	36(1,81%)	28(1,41%)	28(1,41%)	34(0,10%)
	FN	24	27	28	26	16
	CC	68	65	64	66	76
	# Dif.	92	92	92	92	92
	Sens.	73,91%	70,65%	69,57%	71,74%	82,61%
	Prec.	66,67%	64,36%	69,57%	70,21%	69,09%
LUNGHI	**FP**	15(2,64%)	10(1,76%)	16(2,82%)	15(2,64%)	2(0,35%)
	FN	41	48	49	47	37
	VP	50	43	42	44	54
CIRC.	**FP**	10(11,36%)	12(13,64%)	7(7,95%)	4(4,55%)	15(17,05%)
	FN	1	1	2	1	1
	VP	11	11	10	11	11
ALTRO	**FP**	9(0,67%)	14(1,05%)	5(0,37%)	9(0,67%)	17(1,27%)
	FN	34	30	32	30	31
	VP	39	43	41	43	42

Tabella 3.24: Risultati dei migliori classificatori ottenuti tramite features di istogrammi per il dataset REV

TIP.FEAT.		ISTOGRAMMI				STAT.
KERNEL		**RBF**	χ^2	L_1	**HIK**	**REF.**
GLOBALE	**FP**	62(6,93%)	40(4,47%)	26(2,91%)	25(2,79%)	45(5,03%)
	FN	37	41	43	43	30
	CC	91	87	85	85	98
	# Dif.	128	128	128	128	128
	Sens.	71,09%	67,97%	66,41%	66,41%	76,56%
	Prec.	59,48%	68,50%	76,58%	77,27%	68,53%
LUNGHI	**FP**	7(3,26%)	9(4,19%)	2(0,93%)	2(0,93%)	4(1,86%)
	FN	181	176	193	193	156
	VP	52	57	40	40	77
CIRC.	**FP**	22(24,72%)	11(12,36%)	9(10,11%)	11(12,36%)	13(14,61%)
	FN	28	41	38	39	30
	VP	71	58	61	60	69
ALTRO	**FP**	33(5,58%)	20(3,38%)	15(2,54%)	12(2,03%)	28(4,74%)
	FN	268	268	274	270	191
	VP	126	126	120	124	203

Tabella 3.25: Risultati dei migliori classificatori ottenuti tramite features di istogrammi per il dataset STD

3.6.8 Settimo Test

Tipologia di Features: Come tipologia di feaures in questo test si è scelto di utilizzare la media e la varianza del blob interno ed esterno. Il vettore features è quindi così composto:

$$featurevector = [MEAN_{int} \quad VAR_{int} \quad MEAN_{ext} \quad VAR_{ext}]$$

Database Training: Database di training A B

Tipologia di Kernel utilizzato: Kernel

Schema di classificazione: Schema 1 e 2

I buoni risultati ottenuti dai nuovi kernel utilizzati nel precedente test, hanno indotto a provare tali kernel anche per le features statistiche. Si vuole provare se, come nel caso precedente, tali kernel portino alcuni vantaggi come: un range minore dei valori e un maggior valore della precisione. Sono stati fatti i test con entrambi i database di test e con entrambi gli schemi di classificazione.
Nella fase di test si è potuto notare ancora una volta, come i migliori risultati per la tipologia REV derivino da classificatori allenati con il primo database di training. Al contrario di quello che accadeva nel quarto test però, per tale tipologia di immagini, i risultati migliori con i nuovi kernel si ottengono con il secondo schema di classificazione tramite il quale migliorano sia la precisione che la sensitività. Al contrario, per la tipologia di immagini STD, i migliori risultati si ottengono con schema 1 e database di training B. Nelle tabelle 3.26 e 3.29 sono riportati i risultati dei migliori classificatori ottenuti per ciascun kernel, mentre nelle tabelle 3.27 e 3.30 i loro parametri. I risultati ottenuti sono stati posti a confronto con il classificatore CL2 del test 1, per la tipologia REV, e test 4, per la tipologia STD, classificatori che avevano ottenuto in quei test il più alto valore di sensitività. Dalle tabelle si può notare come alcune delle osservazioni fatte nel test precedente siano ancora valide. I classificatori con kernel L_1 e HIK risultano ancora i classificatori con il parametro di precisione più alto. Il classificatore con kernel χ_2 risulta nuovamente il classificatore con sensitività maggiore. Tali valori però sono ancori distanti dai migliori risultati ottenuti tramite kernel RBF. I migliori risultati per la tipologia REV si erano ottenuti nel test 2 dove media e varianza del background esterno al blob si erano calcolati su un bounding box allargato orizzontalmente e verticalmente di 64 pixel. Si è voluto applicare le stesse features ai classificatori di tabella 3.27 per vedere se anche i risultati dei classificatori con i nuovi kernel traessero vantaggio dalle nuove features. I risultati sono riportati in tabella 3.28. Tutti i kernel traggono vantaggio dalle nuove features infatti, per tutti si ha un miglioramento della precisione di almeno il 10%. In termini di sensitività l'unico a non trarre vantaggio è il classificatore con kernel L_1 per il quale non si hanno miglioramenti. É interessante guardare i valori dei veri positivi dei sottoclassificatori del kernel χ^2. Per tale classificatore si ottiene un miglioramento delle classificazioni corrette di 5 unità al quale però non corrisponde un aumento uguale o maggiore dei veri positivi. Al contrario per il classificatore HIK il miglioramento sembra totalmente dovuto al sottoclassificatore di difetti lunghi. Nel secondo test si era evidenziato come l'aumento delle classificazioni corrette era dovuto al fatto che tramite un allargamento del bounding box si riuscivano a catalogare in maniera corretta la gran parte dei difetti a S. Dalla visione dell'output dei nuovi classificatori, non si ha invece un miglioramento dovuto ad un difetto particolare. Sia per il kernel RBF che per i nuovi kernel a beneficiare di tale allargamento risulta sempre essere il sottoclassificatore dei difetti lunghi.

		χ^2	L_1	**HIK**	**RBF**
GLOBALE	**FP**	48(2,41%)	16(0,80%)	27(1,36%)	55(2,76%)
	FN	23	29	29	24
	CC	69	63	63	68
	# Difetti	92	92	92	92
	Sensitività	75,00%	68,48%	68,48%	73,91%
	Precisione	58,97%	79,75%	70,00%	55,28%
LUNGHI	**FP**	19(3,35%)	7(1,23%)	11(1,94%)	
	FN	41	50	51	
	VP	50	41	40	
ALTRO	**FP**	29(2,04%)	9(0,63%)	16(1,13%)	
	FN	30	33	33	
	VP	55	52	52	

Tabella 3.26: TEST 7,Risultati dei migliori classificatori ottenuti per la tipologia REV.

	χ^2	L_1	**HIK**
LUNG.	DAT A $\gamma = 3$ C $=2^7$	DAT A $\gamma = 10$ C$=2^0$	DAT A C$=2^5$
ALTRO	DAT A $\gamma = 3$ C $=2^7$	DAT A $\gamma = 1$ C$=10$	DAT A C$=2^5$

Tabella 3.27: TEST 7, Parametri dei migliori classificatori per la tipologia REV. Lo schema di classificazione utilizzato è lo schema due.

		χ^2	L_1	**HIK**
GLOBALE	**FP**	35(1,76%)	7(0,35%)	15(0,75%)
	FN	18	30	21
	CC	74	62	71
	# Difetti	92	92	92
	Sensitività	80,43%	67,39%	77,17%
	Precisione	67,89%	89,86%	82,56%
LUNGHI	**FP**	8(1,41%)	1(0,18%)	6(1,06%)
	FN	41	51	39
	VP	50	40	52
ALTRO	**FP**	27(1,90%)	6(0,42%)	9(0,63%)
	FN	29	36	36
	VP	56	49	49

Tabella 3.28: Test 7, migliori classificatori ottenuti con features statistiche e larghezza del bounding box aumentata di 64 pixel

		χ^2	L_1	**HIK**	**RBF**
GLOBALE	**FP**	44(4,92%)	29(3,24%)	17(1,90%)	44(4,92%)
	FN	38	39	42	37
	CC	90	89	86	91
	# Dif.	128	128	128	128
	Sens.	70,31%	69,53%	67,19%	71,09%
	Prec.	67,16%	75,42%	83,50%	67,41%
LUNGHI	**FP**	9(4,19%)	5(2,33%)	4(1,86%)	17(7,91%)
	FN	181	189	194	176
	VP	52	44	39	57
CIRC.	**FP**	29(32,58%)	13(14,61%)	8(8,99%)	12(7,91%)
	FN	26	43	45	29
	VP	73	56	54	70
ALTRO	**FP**	6(1,02%)	11(1,86%)	5(0,85%)	15(2,54%)
	FN	258	258	288	257
	VP	136	136	106	137

Tabella 3.29: TEST 7,Risultati dei migliori classificatori ottenuti per la tipologia STD.

	χ^2	L_1	**HIK**
LUNGHI	DAT B $\gamma = 3$ C $=2^7$	DAT B $\gamma = 10$ C$=10$	DAT B C$=2^5$
CIRC.	DAT B $\gamma = 3$ C $=2^7$	DAT B $\gamma = 0.1$ C$=10$	DAT B C$=10$
ALTRO	DAT B $\gamma = 3$ C $=2^7$	DAT B $\gamma = 10$ C$=10$	DAT B C$=2^5$

Tabella 3.30: TEST 7, Parametri dei migliori classificatori per la tipologia STD. Lo schema di classificazione utilizzato è lo schema uno

Capitolo 4

Secondo Problema

In questo capitolo si presenterà il classificatore implementato per risolvere il secondo problema, si darà uno schema generale del sistema implementato e si definirà che cosa sono gli Histogram of Oriented Gradients (HOG). Infine si presenteranno i risultati della fase di test.

4.1 Schema del controllore di qualità

Nel secondo problema si vuole determinare se sull'oggetto controllato siano rimaste delle tracce di resina, utilizzata per fissarlo all'interno del tubo da dove viene catturata l'immagine.
L'immagine ottenuta è un immagine rgb di dimensioni 1944×2552. Come detto nel primo capitolo, a tale immagine vengono applicati dei filtri al fine di ottenere l'immagine dell'apertura polare dell'oggetto che si vuole esaminare. Quando l'oggetto risulta *pulito*, quest'ultima immagine presenterà delle fasce orizzontali di diverso colore, mentre se sull'oggetto è presente della resina tale fasce risultano interrotte. Si vogliono individuare i residui di resina e segnalarli sull'immagine. Si utilizzeranno quindi due tecniche utilizzate nella detection object:

- per ottenere i candidati si utilizzerà la tecnica *sliding windows*. I candidati saranno ottenuti andando a scorrere l'immagine prima orizzontalmente e poi verticalmente tramite una finestra di dimensioni fisse. Ogni sotto-immagine così ottenuta sarà analizzata e classificata per determinare o meno la presenza di resina.
- Come feature vector si utilizzeranno gli Histogram of Oriented Gradients (HOG) di ciascun candidato precedentemente individuato.

Presentiamo ora gli Histogram of Oriented Gradients e successivamente si analizzerà la struttura globale del classificatore.

4.2 Histogram of Oriented Gradients Features (HOG Features)

L'istogramma di gradienti orientati è un descrittore di caratteristiche utilizzato per la prima volta da Navneet Dalal e Bill Triggs, ricercatori dell'Istituto Nazionale di Ricerca francese (INRIA), mentre studiavano il problema del rilevamento di pedoni in immagini statiche. L'idea alla base degli HOG è che ogni oggetto o forma contenuto all'interno di un immagine può essere descritto da una distribuzione dell'intensità del gradiente o della direzione degli edge. Il calcolo di queste features è ottenuto dividendo l'immagine in piccole sotto-aree chiamate *celle*. Per ogni cella verrà calcolato l'istogramma dei gradienti. La combinazione di tali istogrammi rappresenta l'HOG features.
Nella fase implementativa si è deciso di utilizzare per calcolare l'HOG fature la funzione **vl_hog** della libreria **VLFeat**, una libreria open source implementata in C che può essere utilizzata anche in MATLAB. Tale funzione basa il calcolo dell'HOG sull'articolo [1] scritto da Dallal e Triggs,

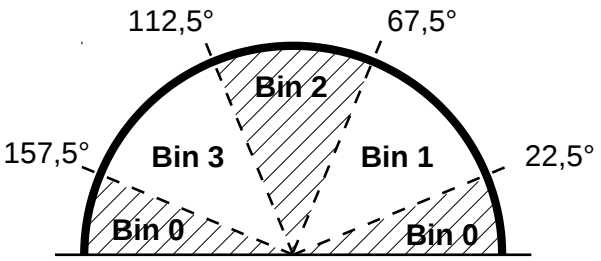


Fig. 4.1: Esempio intervalli bins con contrast insensitive e $p = 4$

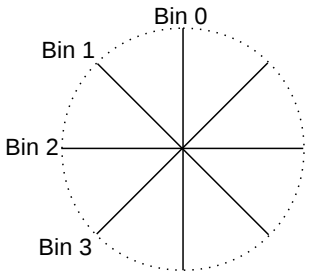


Fig. 4.2: Esempio di visualizzazione HOG con $p = 4$

presenta però anche una seconda modalità che andremo a descrivere, presentata in [2], che non varia il modo di calcolare gli istogrammi bensì presenta una tecnica che riduce le dimensioni della features mantenendo la stessa informazione.

Calcolo Gradiente Il primo step per il calcolo di HOG di un immagine I di dimensioni $w \times h$, è il calcolo di $\theta(x, y)$ e $r(x, y)$, rispettivamente l'orientazione ed il modulo del gradiente del pixel (x,y). Tali valori vengono determinati applicando all'immagine in scala di grigi o ad ognuno dei tre canali di un'immagine RGB, i filtri con kernel $D_x = [-1, 0, 1]$ e $D_y = [-1, 0, 1]^T$. Facendo la convoluzione dell'immagine I con i filtri riportati, si ottengono la derivata dell'immagine lungo x ($I_x = I * D_x$) e lungo y ($I_y = I * D_y$), con le quali si ricavano modulo e orientazione del gradiente :

- (modulo) $r = \left(I_x^2 + I_y^2\right)^{\frac{1}{2}}$
- (direzione) $\theta = \arctan(\frac{I_y}{I_x})$

La direzione del gradiente viene discretizzata in p intervalli la cui ampiezza varia nel caso in cui si voglia rimanere sensibili al contrasto dell'immagine , (B_1), o meno ,(B_2):

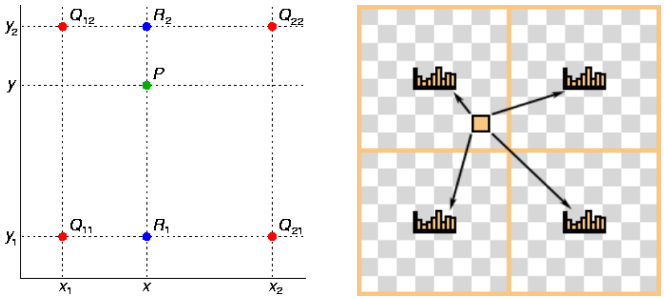
- (Contrast sensitive) $B_1 = round\left(\frac{p\theta(x,y)}{2\pi}\right) mod\,(p)$
- (Contrast insensitive) $B_2 = round\left(\frac{p\theta(x,y)}{\pi}\right) mod\,(p)$

In Figura 4.1 è riportato un esempio degli intervalli dei bins nel caso di $p = 4$ e di contrast insesitive. Nelle fasi successive indichiamo con B sia che si intenda B_1 che B_2.
Si vuole definire ora una *level-pixel feature maps* ossia un istogramma del *modulo* del gradiente per ciascun pixel. Sia $b \in \{0, 1, \ldots, p-1\}$ l'insieme dei valori assunti da B. I bins di ogni pixel (x, y) sono così definiti:

$$F(x,y)_b = \begin{cases} r(x,y) & \text{se} \quad b = B(x,y) \\ 0 & \text{altrimenti} \end{cases} \tag{4.1}$$

Possiamo pensare ad F come ad una *oriented edge map* con p canali di orientazione dove per ciascun pixel si seleziona un canale di discretizzazione tramite l'orientazione del gradiente.

Pixel Aggregation Invece di definire una feature maps per ogni pixel dell'immagine, si divide quest'ultima tramite una griglia, che la divide in quadrati di lato k>0, al fine di rendere i features vector invarianti a piccole deformazioni e di ridurre la dimensione della features map. Si andrà a definire un'unica feature map C(i,j), con $0 \leq i \leq \lfloor (w-1)/k \rfloor$ e $0 \leq j \leq \lfloor (h-1)/k \rfloor$, detta *cell-based feature map*, andando ad aggregare tutte le pixel-based feature map dei pixel



(a) (b)

Fig. 4.3: Interpolazione bilineare

appartenenti alla cella. Il modo più semplice per ottenere la cell-based feature map risulta sommare o fare la media di tutte le level-pixel feature map dei pixel che compongono la cella. Il metodo che viene invece utilizzato dalla funzione *vl_hog* prevede che ogni pixel contribuisce alla formazione dell'istogramma delle quattro celle vicine al pixel utilizzando un'interpolazione bilineare. Un esempio è riportato in Figura 4.3. Sia P il pixel considerato. E siano Q_{ij} con $i, j \in \{1, 2\}$ i centri delle celle di Figura 4.3(b), con coordinate (x_i, y_j). La larghezza delle celle è data quindi da $y_2 - y_1 = x_2 - x_1$. Siano G_{ij}^P i fattori con cui il pixel P contribuisce all'istogramma della cella (i, j), allora sono così definiti:

$$G_{ij}^P = \frac{|x - x_i|}{x_2 - x_1} \frac{|y - y_j|}{y_2 - y_1}$$

Normalizzazione e Troncamento Si vuole minimizzare l'impatto sul descrittore dei cambiamenti di luminosità o del contrasto rispetto al background dell'oggetto ricercato. Ciò si realizza tramite normalizzazione. La funzione usa quattro differenti fattori di normalizzazione per $C(i, j)$. Tali fattori sono indicati con $N_{\delta,\gamma}(i, j)$ dove $\delta, \gamma \in \{1, -1\}$:

$$N_{\delta,\gamma}(i,j) = \left(\|C(i,j)\|^2 + \|C(i+\delta,j)\|^2 + \|C(i,j+\gamma)\|^2 + \|C(i+\delta,j+\gamma)\|^2\right)^{\frac{1}{2}}$$

ognuno dei quali misura l'energia del gradiente dei quattro blocchi quadrati di dimensioni 2×2 contenenti $C(i, j)$. Si indichi con $T_\alpha(v)$ un vettore in cui l'i-esimo componente risulti il valore minimo tra α e l'i-esimo componente di v. L'HOG feature è ottenuta concatenando il risultato della normalizzazione della feature map C rispetto a ciascuno dei 4 fattori $N_{\delta,\gamma}(i, j)$ seguiti dal troncamento:

$$H(i,j) = \begin{bmatrix} T_\alpha(C(i,j)/N_{-1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{-1,+1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,+1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,-1}(i,j)) \end{bmatrix} \tag{4.2}$$

Riduzione dimensioni feature Nel caso in cui si scelga un p pari a 9 si ottiene un HOG a 36 dimensioni. Tale HOG è definito usando nove intervalli di discretizzazione e 4 differenti normalizzazioni. Si ottengono quindi 36 matrici di dimensione $\lfloor (w-1)/k \rfloor \times \lfloor (h-1)/k \rfloor$. Si può pensare ad HOG feature, quindi, come ad una matrice 4×9 che chiameremo *matrice di rappresentazione*. L'autovalore massimo (top eingevector) di ogni componente della matrice di rappresentazione risulta approssimativamente costante lungo ciascuna riga di tale matrice o lungo le colonne. Tali top eingevector appartengono ad un sottospazio lineare definito attraverso dei sparse vector contenenti uni solo lungo una singola riga o colonna della matrice di

rappresentazione. Definiamo $V = \{u_1, u2, \ldots, u_9\} \cup \{v_1, \ldots, v_4\}$ con:

$$u_k(i,j) = \begin{cases} 1 & se \quad j = k \\ 0 & altrimenti \end{cases} \quad e \quad v_k(i,j) = \begin{cases} 1 & se \quad i = k \\ 0 & altrimenti \end{cases} \tag{4.3}$$

Se consideriamo l'esempio iniziale, cioè p pari a nove, possiamo definire un vettore di dimensione 13 andando a considerare il prodotto scalare tra le 36 matrici che costituiscono l'HOG feature con ciascun v_k e u_k. La proiezione di u_k è calcolata sommando le 4 normalizzazioni per una orientazione fissata. La proiezione di v_k è calcolata andando a sommare le nove orientazioni tenendo fissa la normalizzazione.
Del vettore così ottenuto 9 features riguardano l'orientazione del gradiente, mentre 4 misurano l'energia del gradiente in 4 diverse zone nell'intorno della cella considerata.
Alcune applicazioni ottengono migliori risultati utilizzando il contrast sensitive oltre al contrast insesitive. Per tale motivo *vl_hog* le usa entrambe. Sia $C(i,j)$ una cell-based feature maps calcolata aggregando le pixel-level feature maps con nove inesitive contrast orientazioni. Sia $D(i,j)$ una cell-based feature maps calcolata aggregando le pixel level feature maps con 18 contrast sensitive orientazioni. Usiamo come fatto in precedenza 4 fattori di normalizzazione. Normalizzando $C(i,j)$ e $D(i,j)$ e troncando si ottiene un feature vectors $F(i,j)$ di dimensione $4(9+18) = 108$. Applicando la riduzione precedentemente descritta si ricava una feature map di dimensioni ridotte pari a $18 + 9 + 4 = 31$. In definitiva sia p il numero di bin in cui si vuole discretizzare la direzione del gradiente, le feature map avranno avranno dimensione $4 + 3d$.

La funzione *vl_hog* utilizzata necessita in ingresso del numero di pixel del lato di ogni cella (parametro k), degli intervalli di discretizzazione della direzione del gradiente (parametro p) ed infine della tipologia di HOG che si vuole calcolare, ossia l'HOG con dimensioni originali, presentato nell'articolo [1] o quello di dimensioni ridotte [2]. Nella fase di test verrà utilizzata la seconda tipologia. La stessa funzione restituisce inoltre un'immagine che rappresenta l'HOG feature. In tale immagine l'istogramma di ogni cella viene visualizzato come un raggiera dove i raggi, tra loro equispaziati, rappresentano i bins dell'istogramma(il primo intervallo è rappresentato dalla linea verticale) mentre l'intensità del raggio rappresenta il numero di eventi che cadono nel bin. In Figura 4.2 è riportato un esempio nel caso di $p = 4$.

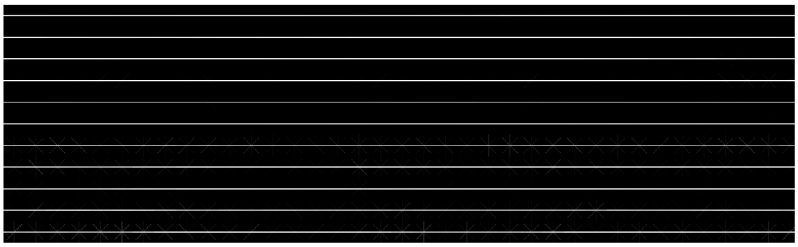
In Figura 4.4 sono stati riportati degli esempi di HOG feature ottenuti con p pari a 4 e k uguale a 12.

L'immagine ottenuta in uscita dal blocco image processing risulta, come già affermato, un'immagine a fasce. La direzione del gradiente è nella maggior parte dei casi verticale, con un angolo di $\pm 90°$. Ciò è confermato dalla Figura 4.4(a). Se sull'oggetto è presente resina, alcune parti dell'immagine presenteranno aree in cui il gradiente non avrà più orientazione verticale ma bensì diverse direzioni, Figura 4.4(b).

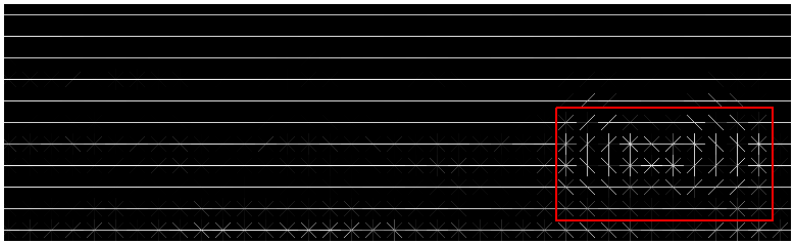
4.3 Detection

Lo schema dell'algoritmo di detection è riportato in Figura 4.5. Per determinare le aree in cui è presente la resina si sfrutterà la differenza tra le HOG features degli oggetti con e senza resina. Per far ciò si è implementata una sliding windows direttamente sul HOG features utilizzando una finestra di dimensioni 3×3, celle scorrendo l'immagine orizzontalmente o verticalmente con passo di una cella. Tali finestre sono i candidati di questo problema e i feature vectors utilizzati sono formati concatenando gli istogrammi di ogni finestra. Tali vettori sono quindi l'input del classificatore implementato tramite SVM.

Definiamo in maniera più rigorosa i termini usati finora:



(a) oggetto pulito



(b) oggetto con presenza di resina

Fig. 4.4: Esempi di sezioni di HOG features

- $I^{(i)}$: i-esima immagine del dataset di test e $y^{(i)}$ la label dell'immagine(1 difetto,-1 non difetto);
- $C^{(i)} = \{c_1^{(i)}, c_2^{(i)}, \ldots, c_n^{(i)}\}$: insieme dei candidati dell'immagine $I^{(i)}$;
- sia $x_j^{(i)}$ l'HOG feature relativo al j-esimo candidato dell'i-esima immagine e sia $\widehat{y}_j^{(i)}$ output del classificatore rispetto a tale candidato
- la label di classificazione $\widehat{y}^{(i)}$ dell'immagine $I^{(i)}$ sarà così definita:

$$\widehat{y}^{(i)} = \widehat{y}_1^{(i)} | \widehat{y}_2^{(i)} | \ldots | \widehat{y}_n^{(i)}$$

allora si ha :

Vero positivo (VP) se data l'immagine $I^{(i)}$ con $y^{(i)} = 1$ allora:

$$\widehat{y}^{(i)} = 1$$

Vero negativo (VN) se data l'immagine $I^{(i)}$ con $y^{(i)} = -1$ allora:

$$\widehat{y}^{(i)} = -1$$

Falso negativo (FN) se data l'immagine $I^{(i)}$ con $y^{(i)} = 1$ allora:

$$\widehat{y}^{(i)} = -1$$

Falso positivo (FP) se data l'immagine $I^{(i)}$ con $y^{(i)} = -1$ allora:

$$\widehat{y}^{(i)} = 1$$

Ai fini della classificazione finale, in questo problema, si considera l'intera immagine. Ad esempio, avremo un vero positivo se nell'oggetto è presente della resina e se almeno uno dei candidati viene classificato come difetto, al contrario se nessun candidato viene classificato come difetto si otterrà un falso negativo.

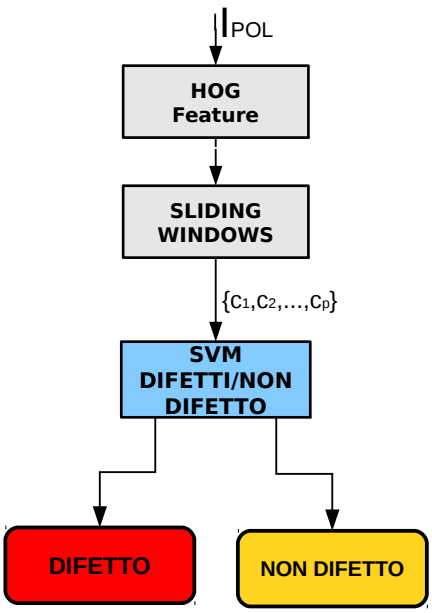


Fig. 4.5: Schema procedura algoritmo di detection del difetto

4.3.1 Training Classificatore

Il database di immagini a disposizioni è composto da un totale di 61 immagini così ripartite: 17 con presenza di resina e 44 senza resina. Per il training del classificatore si sono usate tre immagini, due con difetto e una senza. Da esse si sono ricavate 205 training example di cui 20 appartenenti alla positive class e 185 appartenenti alla negativie class. I features dati in ingresso al classificatore sono, come già detto, la concatenazione degli istogrammi di finestre quadrate di dimensione 3. Poiché si è scelto di discretizzare la direzione del gradiente in quattro intervalli, tali vettori hanno dimensioni $9(4 + 3d) = 144$.
Nei test svolti si è visto che i classificatori che nella fase di test ottenevano i migliori risultati, risultavano quelli con una percentuale di misclassificazioni non inferiore al 2%. I classificatori che utilizzeremo avranno quindi una accuratezza intorno al 98%.

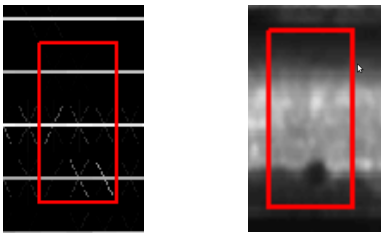
4.4 Test & Risultati

4.4.1 Primo Test

Tipologia di Features: si sono utilizzati degli HOG Feature con celle di dimensione 12 e numero di orientazioni pari a 4. La finestra con cui si determinano i candidati ha dimensione 3×3 celle. Il feature vector è dato dalla concatenazione degli istogrammi delle celle dei candidati.

Tipologia di Kernel utilizzato: Kernel RBF, lineare, polinomiale.

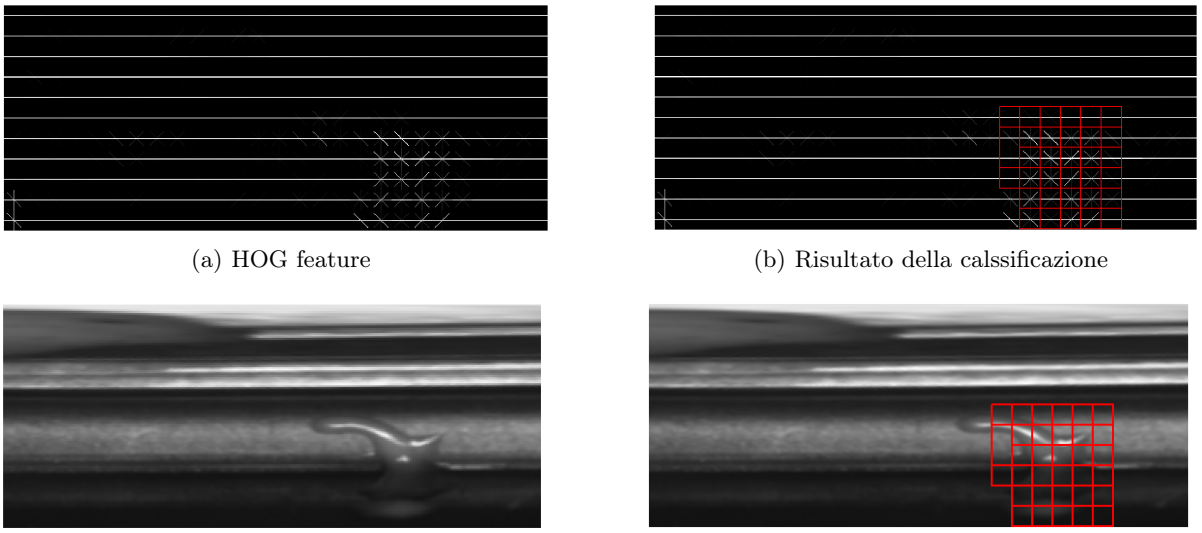
In questo test si sono provati tre classificatori rispettivamente con kernel RBF, polinomiale e lineare i cui parametri sono riportati in tabella 4.2. In tabella 4.1 invece, sono riportati i risultati dei classificatori testati. I risultati sono ottimi per tutte le tipologie di kernel.
Dalla bontà dei risultati del classificatore con kernel lineare si può dedurre che le due classi siano linearmente separabili, come si era ipotizzato nella fase di training. Tale fatto, intersecandolo ai risultati dei classificatori ottenuti, rende superfluo provare a testare in tale problema altri kernel.
In tutti e tre i casi il falso positivo è causato dal candidato riportato in Figura 4.6. Come si può vedere l'immagine del candidato mostra che ciò che si individua non è resina ma una macchia



(a) HOG feature del candidato

(b) Immagine del candidato

Fig. 4.6: Candidato che provoca il Falso Positivo per tutti e tre i classificatori testati



(a) HOG feature

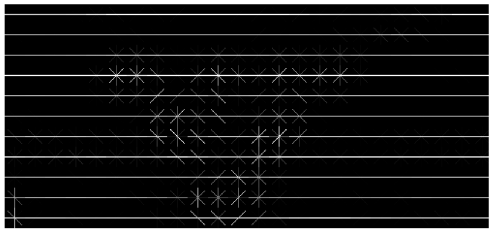
(b) Risultato della calssificazione

(c) Apertura polare della sezione circolare dell'oggetto analizzato

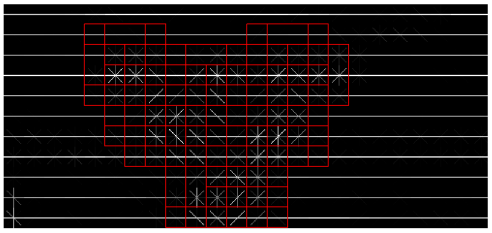
(d) Risultato della classificazione riportato sull'immagine di partenza

Fig. 4.7: Sezione dell'immagine di test M_013

del materiale con cui è prodotto l'oggetto. Non è l'unica immagine che presenta tali macchie, ma è l'unica macchia che viene erroneamente classificata a causa della sua dimensione.
Nelle figure 4.8 e 4.7 invece sono stati riportati alcuni esempi d'identificazione di resina sulla superficie dell'oggetto. Vengono riportate solo alcune sezioni dell'apertura circolare dell'immagine originale e il corrispondente HOG features. Dalle immagini è possibile notare come la resina viene individuata quasi nella sua interezza. Si è provato ad utilizzare finestre di dimensioni più grandi (finestre di dimensioni 4×4 celle e con dimensione del feature vector pari a 256) per vedere se si riuscisse ad evitare tale falso positivo. Si sono utilizzati classificatori con lo stesso kernel del caso precedente ma con parametri del modello diversi, riportati in tabella 4.4. Il training è stato effettuato utilizzando le stesse tre immagini ma acquisendo nuovi features example. I risultati, tabella 4.3, migliorano, raggiungendo una precisione del 100%. Non si esclude che tale miglioramento possa essere dovuto anche da una più accurata scelta dei training example scelti per il training dei classificatori.
Gli ottimi risultati raggiunti con un numero di training example della classe difetti molto piccolo, consigliano di non effettuare altri test. Sarebbe interessante provare a testare gli algoritmi per un dataset di test più numeroso, ma ciò non è stato possibile.

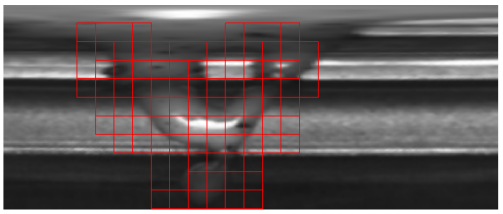


(a) HOG feature



(b) Risultato della calssificazione

(c) Apertura polare della sezione circolare dell'oggetto analizzato



(d) Risultato della classificazione riportato sull'immagine di partenza

Fig. 4.8: Sezione dell'immagine di test D_029

	VP	**FP**	**VP**	**FN**
Polinomiale	15/15	1	42/43	0
Lineare	15/15	1	42/43	0
RBF	15/15	1	42/43	0

Tabella 4.1: Test 1, risultati dei classificatori.

	C	γ	**degree**
Polinomiale	2^{10}		2
Lineare	2^{17}		
RBF	2^{4}	0,05	

Tabella 4.2: Parametri dei classificatori utilizzati nel training del test uno

	VP	**FP**	**VP**	**FN**
Polinomiale	15/15	0	43/43	0
Lineare	15/15	0	43/43	0
RBF	15/15	0	43/43	0

Tabella 4.3: Test 1, risultati dei classificatori.

	C	γ	**degree**
Polinomiale	2^{12}		2
Lineare	2^{17}		
RBF	2^{3}	0,05	

Tabella 4.4: Parametri dei classificatori utilizzati nel training del test uno

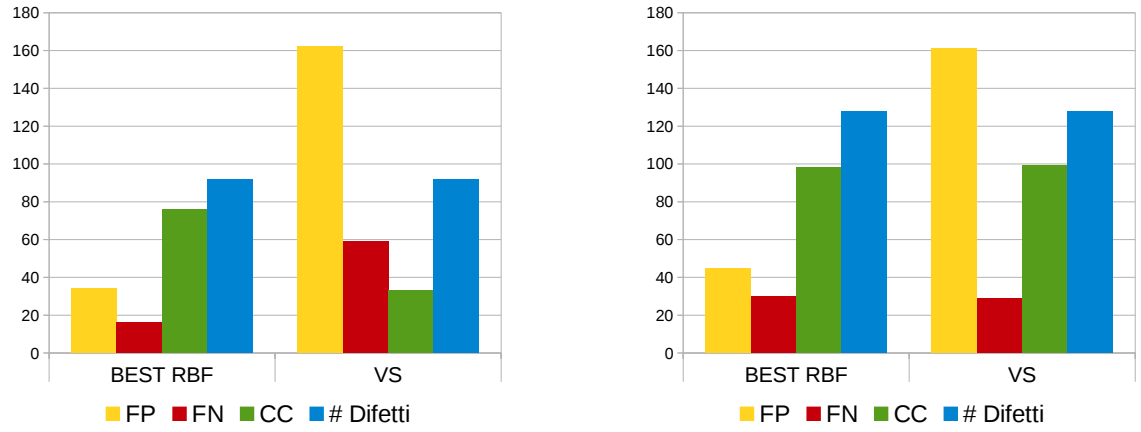
Capitolo 5

Conclusioni e sviluppi futuri

Obiettivo dell'elaborato è lo studio delle Support Vector Machines (SVM) come metodo di apprendimento supervisionato per la classificazione di pattern, al fine di utilizzarle nell'ambito di due problemi applicativi di controllo qualità. Nel secondo capitolo si sono studiate le SVM per mettere in evidenza il significato dei vari parametri e l'effetto che essi hanno ai fini pratici.
Il primo problema affrontato riguarda il controllo qualità del fondo di un cilindro metallico. Lo scopo è determinare e segnalare sull'immagine la presenza di difetti o di scarti di lavorazione. Il problema risulta di difficile approccio per la varietà di difetti e di tipologia del fondo. Si è utilizzata una tecnica di blob detection per determinare le regioni di interesse delle immagini, definite candidati. Il classificatore implementato ha una struttura ad albero in cui i candidati vengono inizialmente divisi per forma, per poi essere classificati in classi difetto/non difetto. Durante i test si sono utilizzate due tipologie di feature vectors: descrittori statistici del colore del candidato e istogrammi del colore, di cui si sono fatti variare i bins. Per entrambi i vettori si sono effettuati test facendo variare lo schema di classificazione, le dimensioni del bounding box del candidato, la funzione di kernel e i parametri. Dai test svolti e presentati nel terzo capitolo, si sono determinati i migliori classificatori (tabella 3.16). Per migliore si intende quel classificatore che possiede il più alto valore di sensitività con una precisione di almeno il 60%. Si sono inoltre potute raccogliere alcune considerazioni finali:

- i migliori risultati sono ottenuti con feature vector di descrittori statistici del colore e una funzione kernel RBF, tabella 3.16;
- i classificatori che utilizzano feature vector di descrittori statistici hanno risultati migliori rispetto a classificatori allenati con feature vector di istogrammi;
- i classificatori con valore di sensitività maggiore si ottengono con kernel χ^2 ed RBF. Il primo però necessita di una fase di tuning meno gravosa;
- i migliori classificatori per la tipologia REV si ottengono con i descrittori estratti dai candidati con bounding box allargato orizzontalmente e verticalmente di 64 pixel. Il classificatore che da questa tipologia di descrittori trae più giovamento è il sottoclassificatore di difetti lunghi;
- il classificatore con kernel HIK è il classificatore con valori di precisione più alti;
- nei diversi test svolti, i classificatori con funzione di kernel laplaciana non sono mai risultati classificatori con valori di sensitività o precisione elevati. Per questo motivo, nel caso di sviluppi futuri si consiglia di non prenderlo in esame.

In Figura 5.1 sono riportati i risultati dei classificatori che presentano sensitività maggiore, ponendoli a confronto con il classificatore di Video Systems. Le migliorie ottenute sono evidenti: i falsi positivi della soluzione presentata risultano un quarto rispetto al classificatore di Video



(a) REV

(b) STD

Fig. 5.1: Istogrammi di confronto tra i migliori classificatori ottenuti e il classificatore Video Systems

Systems, in corrispondenza ad un numero uguale (STD) o doppio (REV) di classificazioni corrette. Le percentuali di sensitività risultano ancora troppo basse per poter utilizzare la soluzione implementata in un'applicazione industriale. Per tali motivi, i risultati ottenuti possono essere considerati come un punto di partenza e non d'arrivo.

Il secondo problema applicativo è il controllo di qualità di oggetti circolari, utilizzati come intramezzo in un cilindro metallico. Si vuole individuare la presenza o meno di residui di resina. Rispetto al primo, tale problema risulta più semplice in quanto la resina è l'unico difetto che si può presentare. Come soluzione si è deciso di sfruttare la particolare composizione del gradiente dell'immagine dell'apertura polare dell'oggetto tramite le HOG features. Tali descrittori sono utilizzati come input dei classificatori, implementati tramite SVM, mentre i candidati sono precedentemente individuati con la tecnica delle sliding windows. I descrittori HOG si sono dimostrati degli ottimi descrittori per l'analisi di qualità superficiali. Nel quinto capitolo si è evidenziato che in fase di test viene raggiunto il 100% in precisione e sensitività. Le percentuali ottenute hanno permesso di testare il programma direttamente in linea di produzione, fornendo ottimi risultati, di cui però non si è in grado di dare valori numerici.
L'utilizzo delle SVM nei due problemi pratici ha permesso di individuare alcuni punti di forza delle Support Vector Machines, ossia:

- la chiarezza del significato dei parametri con cui si allenano i classificatori;
- il kernel trick. Le diverse funzioni kernel provate nel primo problema hanno mostrato come ogni funzione porti a risultati diversi, ognuna con le proprie specificità;
- la semplicità di implementazione.

Le SVM presentano però alcuni svantaggi: la difficoltà nel determinare la tipologia di descrittori che forniscono le migliori prestazioni, la dispendiosità in termini di tempo della fase di tuning e la difficoltà nel trovare il giusto trade-off tra le features e la funzione kernel utilizzate. Tali problematiche, intrinseche alle SVM, portano, nel caso di problemi complessi come il primo, ad una fase di test molto dispendiosa in termini di tempo, tempo che a volte in azienda non sempre si ha a disposizione.
Il lavoro svolto può quindi essere sviluppato con l'obiettivo di migliorare le prestazioni temporali. La letteratura offre alcune possibili soluzioni come, ad esempio, le Multiple Kernel Learning

(MKL). Tale tecnica combina in maniera lineare i risultati delle diverse funzioni kernel utilizzate ognuna delle quali può presentare features vector diversi e parametri diversi e ottimizza i parametri della combinazione, al fine di ottenere il classificatore ottimo. Tale tecnica migliora tutti i punti deboli delle SVM: il tuning viene svolto in maniera automatica per una parte del problema e permette di utilizzare diverse tipologie di descrittori, pesati opportunamente da MKL.

In conclusione ringraziamo Video Systems per l'opportunità che mi ha concesso ed in particolare Daniele Fornasier per il tempo dedicato e per la disponibilità.

Appendice A

Problemi di Ottimizzazione Convessa

Definizione A.0.1. Problema di ottimizzazione convessa
Sia dato il Problema:

$$\begin{aligned} p^\star = \min \quad & f(x) \\ & g_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_j(x) = 0 \quad j = 1, \ldots, p \end{aligned} \tag{A.1}$$

Si assuma che i vincoli di disuguaglianza siano dati da funzioni $g_i(x)$ convesse in $\mathbb{R}^n$ e che i vincoli di uguaglianza siano dati da funzioni $h_i(x)$ affini. Se la funzione obiettivo $f(x)$ è una funzione obiettivo in $\mathbb{R}^n$, il Problema è detto di ottimizzazione convessa . La variabile x è detta variabile primale del problema ed è considerata ammissibile quando $x \in D$ dove D è il dominio del problema $D = \bigcap_{i=1}^{m} dom\, g_i(x) \cap \bigcap_{i=1}^{p} dom\, h_i(x)$.

Nota. Per un problema di ottimizzazione convessa, un punto di minimo locale è un punto di minimo globale.

Definizione A.0.2. Lagrangiana
Definiamo la Lagrangiana del problema (A.1) come la funzione $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$:

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{l} \beta_i h_i(x)$$

dove le variabili α e β sono dette variabili duali o moltiplicatori di Lagrange e il dominio $L = D \times \mathbb{R}^m \times \mathbb{R}^p$.

Definiamo come problema primale il seguente problema :

$$p^\star = \min_x \max_{\alpha,\beta:\alpha\geq 0} L(x, \alpha, \beta)$$

Tale problema risulta uguale al problema principale (A.1) se:

- $g_i(x) > 0$ per qualche i allora $\max_{\alpha,\beta:\alpha\geq 0} L(x, \alpha, \beta) = \infty$
- $h_i(x) \neq 0$ per qualche i $\max_{\alpha,\beta:\alpha\geq 0} L(x, \alpha, \beta) = \infty$
- se x assume un valore ammissibile dai vincoli allora risulterà $\max_{\alpha,\beta:\alpha\geq 0} L(x, \alpha, \beta) = f(x)$

L'idea è quella di inserire delle penalità per far sì che nella risoluzione di tale problema vengano prese in considerazione solo delle variabili ammissibili. Sotto queste condizioni il problema iniziale è uguale al problema primale e, quindi, sarà usato quest'ultimo termine per indicare entrambi i quesiti.

Definizione A.0.3. Funzione Duale di Lagrange

Definiamo la Funzione Duale $\theta_D : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ come il minimo valore della lagrangiana su x per $\alpha \in \mathbb{R}^m$ e $\beta \in \mathbb{R}^p$:

$$\theta_D(\alpha, \beta) = \min_{x \in D} L(x, \alpha, \beta)$$

Nota. La Funzione Duale rappresenta un lower bound per il risultato ottimo del problema primale $\theta(\alpha, \beta) < p^\star$.

Dimostrazione: Sia $\bar{x}$ un punto ammissibile. Allora $g_i(\bar{x}) \leq 0$ e $h_i(\bar{x}) = 0 \ \forall i = 1, \ldots, m$ e $\alpha \geq 0$:
$\sum_{i=1}^m \alpha_i g_i(\bar{x}) + \sum_{i=1}^l \beta_i h_i(\bar{x}) \leq 0$
Quindi

$$\theta_D(\alpha, \beta) = \min_{x \in D} L(x, \alpha, \beta) \leq L(\bar{x}, \alpha, \beta) = f(\bar{x}) + \sum_{i=1}^{m} \alpha_i g_i(\bar{x}) + \sum_{i=1}^{l} \beta_i h_i(\bar{x}) \leq f(\bar{x})$$

per ogni $\bar{x} \in D$ e quindi anche per x ottimale.

La funzione duale Lagrangiana fornisce un limite inferiore del valore ottimale $p^\star$, limite che dipende dai valori di α e β. Vogliamo valutare qual è il miglior valore della funzione duale.

Definizione A.0.4. Problema Duale (Lagrange Dual Problem)

Definiamo problema duale il problema di massimizzazione :

$$\begin{array}{ll} d^\star = \max & \theta_D(\alpha, \beta) \\ s.t. & \alpha \geq 0 \end{array}$$

e siano $(\alpha^\star, \beta^\star)$ i valori per cui si ottiene il risultato di ottimo $d^\star$, detti valori ottimi di Lagrange.

Definizione A.0.5. Dualità debole

Sia $d^\star$ il valore ottimo del problema Duale e sia $p^\star$ il valore ottimo del problema Primale. Allora $d^\star \leq p^\star$, cioè la soluzione del problema duale, è il più grande limite inferiore del problema primale. Il valore $p^\star - d^\star$ è detto gap ottimale.

Tale relazione è vera in qualsiasi problema di ottimizzazione anche nel caso di problemi non convessi.

Definizione A.0.6. Dualità Forte

Si considerino il problema primale e il problema duale. Se esiste una soluzione ammissibile del problema primale e una ammissibile per il problema duale tale per cui $p^\star = q^\star$, allora si parlerà di dualità forte. In tal caso il gap ottimale è nullo.

Teorema 2. *Slater's condition*

Se esiste una soluzione ammissibile del problema primale $\hat{x}$ per la quale i vincoli sono strettamente soddisfatti, ossia $g_i(\hat{x}) < 0 \ \forall i = 1, \ldots, m$, allora $p^\star = q^\star$.

Teorema 3. *Siano f e g_i funzioni convesse differenziabili e $h_i(x)$ funzioni affini. Supponiamo che i vincoli $g_i(x)$ siano strettamente rispettati (strictly feasible), ossia che esiste un x tale per cui $g_i(x) < 0$. Allora sotto queste ipotesi esistono un $\hat{x}$ soluzione ammissibile del problema primale e $\hat{\alpha}, \hat{\beta}$ soluzione ammissibile del problema duale tali che $p^\star = d^\star = L(\hat{x}, \hat{\alpha}, \hat{\beta})$. Ed inoltre i valori ottimi soddisfano le condizioni di Karush-Kuhn-Tucker (KKT) :*

$$\begin{array}{lll} (KKT1) & \nabla_x L(\hat{x}, \hat{\alpha}, \hat{\beta}) = 0 & \\ (KKT2) & \hat{\alpha}_i g_i(\hat{x}) = 0 & i = 1, \ldots, m \end{array} \tag{A.2}$$

allora $\hat{x}$ è la soluzione ottima problema primale,$\hat{\alpha}, \hat{\beta}$ sono la soluzione ottima del problema duale e i vincoli del problema primale sono strettamente soddisfatti.

Teorema 4. *Siano f e g_i funzioni convesse e $h_i(x)$ funzioni affini.$(\hat{x}, \hat{\alpha}, \hat{\beta})$ siano dei punti che soddisfano le condizioni Karush-Kuhn-Tucker (KKT) :*

$$\begin{aligned} g_i(\hat{x}) &\leq 0 & i &= 1, \dots, m \\ h_i(\hat{x}) &= 0 & i &= 1, \dots, p \\ \hat{\alpha}_i &\geq 0 & i &= 1, \dots, m \\ \hat{\alpha}_i g_i(\hat{x}) &= 0 & i &= 1, \dots, m \\ \nabla_x L(\hat{x}, \hat{\alpha}, \hat{\beta}) = \nabla f(\hat{x}) + \textstyle\sum_{i=1}^{m} \hat{\alpha}_i \nabla g_i(\hat{x}) + \sum_{i=1}^{l} \hat{\beta}_i \nabla h_i(\hat{x}) &= 0 \end{aligned} \tag{A.3}$$

allora $\hat{x}$ è la soluzione ottima del problema primale, $\hat{\alpha}$ e $\hat{\beta}$ sono la soluzione ottima del problema duale e si ha $p^\star = d^\star = L(\hat{x}, \hat{\alpha}, \hat{\beta})$. La prima e la seconda condizione dicono che $\hat{x}$ è un punto ammissibile, la terza e la quarta dicono che finché $\hat{\alpha}_i \geq 0$ $L(x, \hat{\alpha}, \hat{\beta})$ è convessa e l'ultima dice che il gradiente della funzione Lagrangiana si annulla per $x = \hat{x}$ e che quindi minimizza la funzione in x.

Nota. Per ogni problema di ottimizzazione convessa con funzioni obiettivo e funzioni del vincolo differenziabili, qualsiasi punto che soddisfa le condizioni KKT è punto ottimale con gap di dualità pari a zero.

Nota. Se un problema di ottimizzazione con la funzione obiettivo e le funzioni del vincolo differenziabili soddisfa le condizioni di Slater (e quindi il gap di ottimalità è zero), allora le condizioni KKT sono condizioni necessarie per l'ottimalità.

Appendice B

Librerie utilizzate

Prima di analizzare il codice implementato si vuole fare un breve excursus sulle librerie open source utilizzate. Si sono utilizzati due tool esterni a Matlab **libSVM** e **vlFeat**: il primo utilizzato per allenare i modelli dei classificatori e per classificare le istanze, il secondo per il calcolo delle HOG features.

B.1 LIBSVM

LIBSVM è una libreria open source di machine learning, sviluppata dalla Nationale Taiwan University, per la classificazione, regressione e stima della distribuzione. Il tool è scaricabile dal sito `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. L'obiettivo della libreria è quello di permettere un facile utilizzo delle SVM. Il pacchetto comprende tre funzioni: svmtrain, svmpredict e svmscale. La prima funzione addestra l'SVM attraverso il training set, la seconda per testa l'SVM attraverso un test set e la terza è utilizzata per normalizzare i dati nel range predefinito, in genere $[0, 1]$. Il pacchetto è utilizzabile da linea di comando o è esportabile in altri linguaggi come Java, MATLAB e R, per i quali però non è prevista la funzione svmscale. LIB SVM utilizza gli algoritmi Sequential Minimal Optimization (SMO) per risolver i problemi QP delle SVM. Il tool è in grado di risolvere efficientemente cinque problemi:

- C-Support Vector Classification
- ν-Support Vector Classification
- ϵ-Support VectorRegression
- ν-Support Vector Regression
- Distribution Estimation (on-class SVM)

I passi fondamentali nell'utilizzo del tool sono:

1. convertire i dati in input nel formato del tool che si intende utilizzare;
2. effettuare la normalizzazione dei dati nel range opportuno;
3. scegliere il tipo di funzione kernel da utilizzare;
4. scegliere o determinare i parametri C e i parametri del kernel;
5. dati i valori scelti effettuare il training dell'SVM.

Analizziamo più nel dettaglio le fasi principali.

Training

Il tool fornisce diverse opzioni di training. In primo luogo permette di scegliere la tipologia di problema SVM da risolvere, attraverso l'opzione **-s**. É possibile scegliere tra:

- 0 – C-SVC (multi-class classification) parametro di default
- 1 – ν-SVC (multi-class classification)
- 2 – one-class SVM
- 3 – ϵ-SVR (regression)
- 4 – ν-SVR (regression)

nei nostri problemi si è sempre utilizzata la prima tipologia. Tramite l'opzione **-t** è possibile scegliere tra 5 tipologie di kernel diverse:

- 0 – lineare
- 1 – polinomiale
- 2 – RBF
- 3 – sigmoid
- 4 – kernel precompilato

Legate ai kernel esistono una serie di opzioni che permettono di impostare i parametri desiderati, di seguito riportiamo quelli utilizzati:

- **-d** degree : permette di impostare il grado del kernel polinomiale (default 3)
- **-g** gamma : permette di impostare il gamma nelle funzioni kernel (default 1/num_features)
- **-r** : permette di impostare r nel kernel polinomiale (2.14) (default 0)
- **-c** cost : permette di impostare il parametro C dei problemi C-SVC, epsilon-SVR, and nu-SVR (default 1)

L'invocazione delle funzioni può avvenire tramite linea di comando o richiamandole in matlab. Nel progetto si sono utilizzate entrambe le possibilità. Per utilizzare i kernel appartenenti alla libreria si sono richiamate le funzioni tramite matlab, mentre quando si utilizzavano kernel precompilati, ossia al di fuori dei kernel lineari, polinomiali, RBF e sigmoid, si è utilizzata tramite linea di comando.
Da linea di comando bisogno invocare la funzione:

```
svm-train <opzioni> training_file
```

dove *<opzioni>* sono una successione di caratteri per specificare le opzioni con cui si vuole allenare l'SVM, e *training_file* è un file contenente i training examples con un formato ben specifico che verrà esposto nel prossimo paragrafo. Tale funzione restituisce il file *training_file.model*, contenente i parametri del modello. Ossia:

- Parameters: i parametri
- nr_class: numero si classi
- totalSV: la cardinalità dell'insieme dei support vectors (SV)
- rho: la variabile b (bias) della funzione di decisione wx+b

- Label: le labels delle classi;
- ProbA: pairwise probability information; empty if -b 0 or in one-class SVM
- ProbB: pairwise probability information; empty if -b 0 or in one-class SVM
- nSV:il numero di support vectors per ciascuna classe
- sv_coef: è il valore deigli α associati ai SV
- SVs: i support vectors.

dove ProbA e ProbB sono valori che si ottengono nei problemi di regressione.
Se si usa matlab, invece, la funzione da richiamare è:

```
model = svmtrain(training_label_vector, training_instance_matrix [,
                        'libsvm_options']);
```

dove *training_label_vector* è un vettore di dimensioni $L \times 1$ contenente le labels dei training examples, *training_instance_matrix* è una matrice $L \times n$ contente i feature vectors, ed n è la loro dimensione. *libsvm_options* è una stringa contenente le opzioni con cui si vuole allenare l'SVM mentre *model* è la struttura contenente il modello dell'SVM, che presenta le medesime informazioni del modello allenato da linea di comando.

Formato dei dati

Nel caso si utilizzi il pacchetto LIBSVM da linea di comando il file *training_file*, che bisogna passare in ingresso, ha una struttura ben precisa. Assumiamo che il nostro training set sia formato dall'insieme di training examples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)\}$. E sia $K(x, y)$ il kernel per le istanze x e y. Il formato dei dati per la fase di training dovrà risultare:

$$\begin{array}{cccccc} y_1 & 0:1 & 1:K(x_1,x_1) & 2:K(x_1,x_2) & \ldots & L:K(x_1,x_L) \\ y_2 & 0:2 & 1:K(x_2,x_1) & 2:K(x_2,x_2) & \ldots & L:K(x_2,x_L) \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ y_L & 0:L & 1:K(x_L,x_1) & 2:K(x_L,x_2) & \ldots & L:K(x_L,x_L) \end{array}$$

Mentre per la fase di test rispetto alla quale i valori di label sono sconosciuti il formato dei dati risulterà:

$$\begin{array}{cccccc} \star & 0:? & 1:K(x_1,x_1) & 2:K(x_1,x_2) & \ldots & L:K(x_1,x_L) \\ \star & 0:? & 1:K(x_2,x_1) & 2:K(x_2,x_2) & \ldots & L:K(x_2,x_L) \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ \star & 0:? & 1:K(x_L,x_1) & 2:K(x_L,x_2) & \ldots & L:K(x_L,x_L) \end{array}$$

dove $\star$ risulaterà uno dei valori delle labels associate al problema mentre ? può assumere un qualsiasi valore.

Quello presentato finora è il formato dati necessario quando si utilizza il tool LIBSVM da linea di comando. Per creare i file con tale struttura, si è implementata la funzione **creaFILE(trainlabels,Ktrain,nomefile)**. Tale funzione crea un file, di nome <*nomefile*> all'interno della cartella da cui viene evocata, in cui legge le label, *trainlabels*, e i valori del kernel, *Ktrain*, e crea la struttura dati sopra presentata.

Test Per la fase di test o predizione la funzione da invocare da linea di comando è:

```
svm-predict predict_file training_file.model <output_file>
```

dove predict_file è un file contenente le istanze da predire, col formato appena esposto, *training_file.model* è il file dell'SVM allenato e *<output_file>* è il file contenente le predizioni. Se si usa matlab invece:

```
predicted_label, accuracy, decision_values/prob_estimates]=
svmpredict(testing_label_vector, testing_instance_matrix, model);
```

dove *testing_label_vector* è un array $m \times 1$, se m è il numero di istanze da testare, contenente le labels. Se sono sconosciute ogni componente dell'array può assumere un valore casuale. *testing_instance_matrix* è una matrice $m \times n$ delle m istanze con dimensione n e *model* è il modello dell'SVM allenato.

B.2 VLFEAT

VLFeat è una libreria open source di visione computazionale, che fornisce algoritmi di image understanding e local features extraction e matching. VLFeat è stata scritta in C per renderla efficiente e compatibile tramite interfacci con MATLAB. Questa libreria fornisce una serie di algoritmi molto potenti. Noi l'abbiamo utilizzata per il calcolo dell'HOG features per il quale mette a disposizione la funzione **vl_hog**. Tale funzione in ingresso vuole:

- `cellSize` il numero di pixel che costituiscono il lato delle celle che vanno a dividere l'immagine;
- `'numOrientations'` il numero di orientazioni con cui si vuole discretizzare la direzione del gradiente
- `'variant', 'dalaltriggs'` è l'opzione da aggiungere se si vuole che l'HOG calcolato sia quello di [1]. Di default la funzione calcola l'HOG a dimensioni ridotte [2].

La funzione da richiamare è:

```
hog = vl_hog(im, cellSize, 'verbose','numOrientations',num, 'variant',
                           'dalaltriggs')
```

La stessa funzione general una figura della hog features come spiegato nel capitolo 4. Per ottenerla è necessario richiamare la funzione:

```
imhog = vl_hog('render', hog, 'verbose') ;
  clf ; imagesc(imhog) ; colormap gray ;
```

B.3 Codice Primo Problema

Di seguito verranno presentati le funzioni implementate per i due problemi presentati. I codici saranno dividi in tre sotto paragrafi.

- **Funzioni generali**: quelle funzioni implementate che saranno usate sia per la fase di training che per la fase di test. Un esempio saranno le funzioni che vanno a calcolare le feature dei candidati.
- **Funzioni di training**: il codice scritto esplicitamente per il training dei classificatori. Fanno parte di questo sotto-paragrafo le funzioni e le GUI scritte per ottenere i training example e quelle per il training dei classificatori.
- **Funzioni per il TEST**: tutte le funzioni che permettono di eseguire i test su nuove immagini.

B.3.1 Funzioni generali

Per il primo problema i candidati vengono forniti dal software di Video Systems. Per ogni immagine il programma restituisce una cartella, col nome dell'immagine analizzata, contenente i file:

- *color.png*: l'immagine originale analizzata;
- *gray.png*: l'immagine originale convertita in scala di grigio;
- *filter_00.png* e/o *filter_01.png* : l'immagine normalizzata dell'oggetto da analizzare. Per la tipologia STD viene restituita solo un'immagine contenente la sola sezione circolare dell'oggetto circolare. Nel caso di REV vengono restituite due immagini. La prima restituisce la fascia circolare più esterna dell'oggetto, mentre la seconda la sezione circolare più interna.
- *candidate_XXX.png*: una sequenza di immagini binarizzate che rappresentano uno dei candidati individuati. Tale immagini saranno le maschere da utilizzare sull'immagine originale per ricavare i candidati.
- *result.png*: l'immagine contenente il risultato del classificatore implementato da Video Systems, nella quale i candidati catalogati come difetto sono segnati in rosso, mentre quelli catalogati come non difetto in giallo.

Per calcolare i descrittori dei candidati, sia per le fasi di training che di test, si utilizzeranno le seguenti funzioni:

- **regionprops(BW,properties)** di MATLAB: che restituisce le misure delle proprietà richieste tramite il vettore *properties* per ciascun componente (blob) dell'immagine binarizzata, *BW*. Dove per componente si intende qualsiasi gruppo di pixel connessi di qualsiasi dimensione. Come output restituisce un array di strutture (struct) contenenti le proprietà richieste per ciascun blob individuato.
- **[output] = blob_rgb(I_gray, I_resize, blob, originale)**: tramite tale funzione si ricavano le color feature dei candidati. Dove *I_gray* è l'immagine normalizzata del nostro candidato (filter_00.png o filter_01.png), *I_resize* è l'immagine in scala di grigio del bounding box del candidato riscalata di un determinato fattore, *blob* è la struttura contenente alcune proprietà del candidato da analizzare, *originale* è un booleano che indica se si va ad analizzare l'immagine originale. Gli input *I_resize* e *originale* erano parametri inizialmente inseristi al fine di determinare alcune features del gradiente dell'immagine *I_resize*. Tali descrittori non sono stati però utilizzati.
- **[output] = blob_bw(BW)**: tale funzione determina descrittori geometrici del blob del candidato, BW. Tale funzione incrementa i descrittori individuati da *regionprops* Anche in questo caso alcuni dei descrittori non sono più stati utilizzati.

B.3.2 Fase di Training

Il codice implementativo con cui si svolge la fase di training si può dividere in due sottosezioni: il calcolo dei training example e l'allenamento dei classificatori.

Determinazione dei training examples

Per determinare i training example è necessario raccogliere in un unica cartella, *<directory>*, tutte le cartelle, *<image_name>*, ottenute come output dal software di Video Systems, delle immagini i cui candidati vogliono essere utilizzati come training example. Si sono implementati:

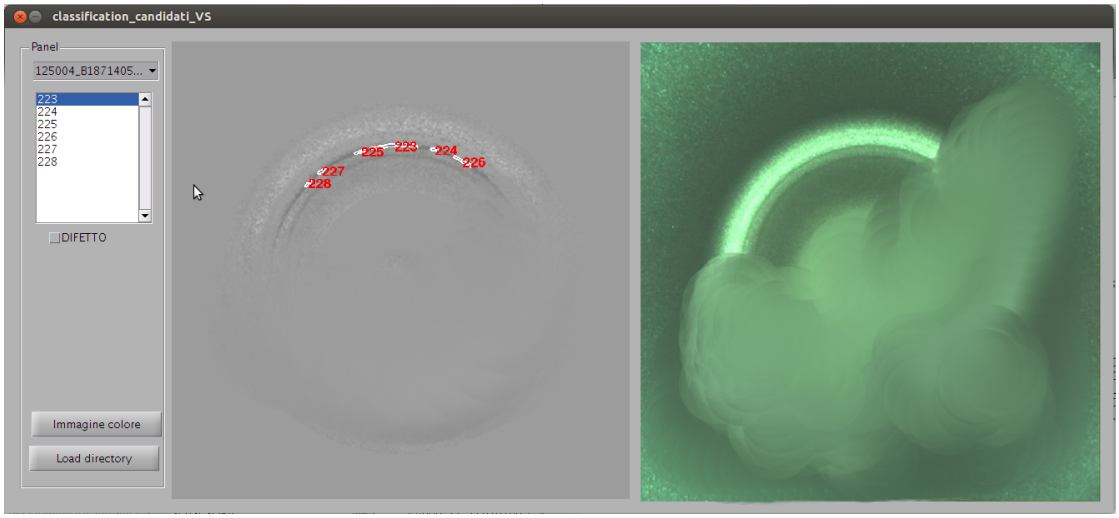


Fig. B.1: Esempio dell'interfaccia GUI classification_candidati_VS utilizzata nella fase di Train

- lo script **Find_candidati** che crea all'interno di *<directory>* due cartelle: *DATI_VS* e *CATALOGAZIONE_VS*. Per ogni cartella *<image_name>*, contenuta in *<directory>* lo script va a determinare le features, attraverso le funzioni presentate precedentemente, dei candidati appartenenti all'immagine *<image_name>*. Per ogni immagine analizzata salva due file. Il file *<image_name>_ftVS.mat*, nella cartella DATI_VS, il quale contiene la struttura con tutte le features dei candidati dell'immagine analizzata, e il file *<image_name>_clVS.mat*, nella cartella CATALOGAZIONE_VS, che contiene una struttura utilizzata per definire le labels dei candidati. Inoltre salva, all'interno della cartella *<image_name>*, le immagini *filter_00_candidati.png* e/o *filter_01_candidati.png*. In tale immagini sono messi in evidenza i candidati e ad ognuno di loro è associato un ID univoco per l'intero dataset di training. Tale immagine viene utilizzata per catalogare i training examples.

- la GUI **classification_candidati_VS**, la cui immagine è riportata in Figura B.1. Tramite la GUI si assegna le label DIFETTO/NON DIFETTO ad ogni candidato del training set. Tramite il pulsante *Load Directory* si selezionerà la cartella *<directory>*. Per catalogare ogni candidato si scorreranno le immagini del training set tramite il menù a tendina e il candidato verrà selezionato tramite la list box. Ogni candidato è individuato sulla prima immagine a sinistra da un ID. Tale immagine è l'immagine *filter_0X_candidati.png* ottenuta dallo script TRAIN.m. Si riporta inoltre una seconda immagine, ossia l'immagine a colori originale. Tramite la check box DIFETTO si assegnerà la label al candidato selezionato sulla list box. Tale catalogazione verrà salvata nel file *<nome_cartella>_clVS.mat*, ogni volta che si seleziona una diversa immagine nel munù a tendina.

- lo script **create_train_feauteres.m**. Dopo aver eseguito il primo script ed aver classificato i candidati tramite la GUI, bisogna eseguire questo script, andando a modificare la variabile *name_dir* con il nome della cartella *<directory>*. Tale script restituisce il file *feature_<directory>.mat* contenente 5 array di strutture che contengono le informazioni e le features dei candidati e una sesta variabile contenente il nome *<directory>*. Tale file .mat sarà utilizzato nella successiva fase di training dei classificatori.

In sostanza, si può dire che, sia (x_i, y_i) un training example appartenente al training set S, allora il training set S è dato dalla cartella *<directory>*, il vettore delle feature, x_i, è salvato nella cartella DATI_SV e la label corrispondente y_i è salvata nella cartella CATALOGAZIONE_VS. Tali training example sono raggruppati in base all'immagine da cui sono stati ricavati.

Training classificatori

Il training dei classificatori viene eseguito in due passi: per prima cosa i candidati vengono suddivisi in base alla loro forma tramite lo script *FORMA_train.m* e poi vengono allenati i singoli sotto-classificatori di difetti tramite gli script *DIFETTI_<forma>.m*. Descrizione degli script:

- *FORMA_train.m*: questo script divide i candidati del file *feature_<directory>.mat* in base alla forma tramite i classificatori SVM salvati nei file *model_CIRCOLARI.mat* e *model_LUNGHI.mat*. In uscita restituisce il file *forma.mat* in cui sono salvate le variabili:
  - *lunghi*: array degli indici dei candidati contenuti nelle struttura *feature_<directory>.mat* che sono stati classificati come *lunghi*
  - *circolari*: array degli indici dei candidati che sono stati classificati come *circolari*
  - *non_catalogati*: array degli indici dei candidati che sono stati classificati come *altro*
  - *catalogazione*: array di dimensione $\#candidati \times 3$ per ogni riga è contenuto un unico uno in base alla forma corrispondente al candidato. Dove la prima colonna corrisponde alla forma lunghi, la seconda alla forma circolare e l'ultima alla forma altro.
- *DIFETTI_<forma>.m*: sono tre script diversi dove <forma> assume i valori CIRCOLARE, LUNGHI, NONCATALOGATI. Tali script hanno la medesima struttura:
  1. definiscono inizialmente il feature vectors dei candidati che appartengono all'insieme <forma>, variabile *trainfeatures*, utilizzando i descrittori contenuti nel file *feature_<directory>.mat*, e i labels corrsipondenti, variabile *trainlabels*;
  2. scalano se necessario i valori di ciascun training example in un range tra 0 e 1. Per far ciò viene utilizzata la formula (3.2) dove m è rappresentata dalla variabile *mi* e M dalla variabile *Mi*;
  3. vengono allenati i classificatori utilizzati tramite la libreria **libSVM**. Il tipo di kernel che si vuole utilizzare per l'allenamento è selezionabile attraverso la variabile *type_kernel* che può assumere i valori:
     - **lineare**, nel caso si voglia utilizzare il kernel lineare;
     - **polinomiale**, nel caso si voglia utilizzare il kernel polinomiale;
     - **RBF**, nel caso si voglia utilizzare il kernel RBF;
     - **HIK**, nel caso si voglia utilizzare il kernel HIK;
     - **L1**, nel caso si voglia utilizzare il kernel laplaciano;
     - **chi**, nel caso si voglia utilizzare il kernel χ^2;

     I valori dei parametri con cui allenare il classificatore vanno modificati direttamente nella funzione `svmtrain` associata al kernel utilizzato.

Come output del blocco di training si ottengono tre file (*model_difetti_lunghi.mat*, *model_difetti_circolari.mat* e *model_difetti_non_catalogati.mat*) nel caso dei primi tre kernel, a cui si aggiungono altri tre file (*DIFETTI_LUNGHI.model*, *DIFETTI_CIRCOLARI.model* e *DIFETTI_NON_CATALOGATI.model*) nel caso si scelga uno degli ultimi tre kernel. Tali file contengono i modelli dei classificatori allenati, i valori di minimo e massimo con cui si sono scalati i feature vector e se necessario i feature vector con cui si è eseguito il training.

Per allenare i classificatori utilizzando il secondo schema di classificazione basta utilizzare il file *File_train_2classi.m* per dividere nelle due forme lunghe e altro i candidati mentre per allenare i classificatori si utilizzeranno solamente i file *DIFETTI_LUNGI.m* e *DIFETTI_NONCATALOGATI.m*

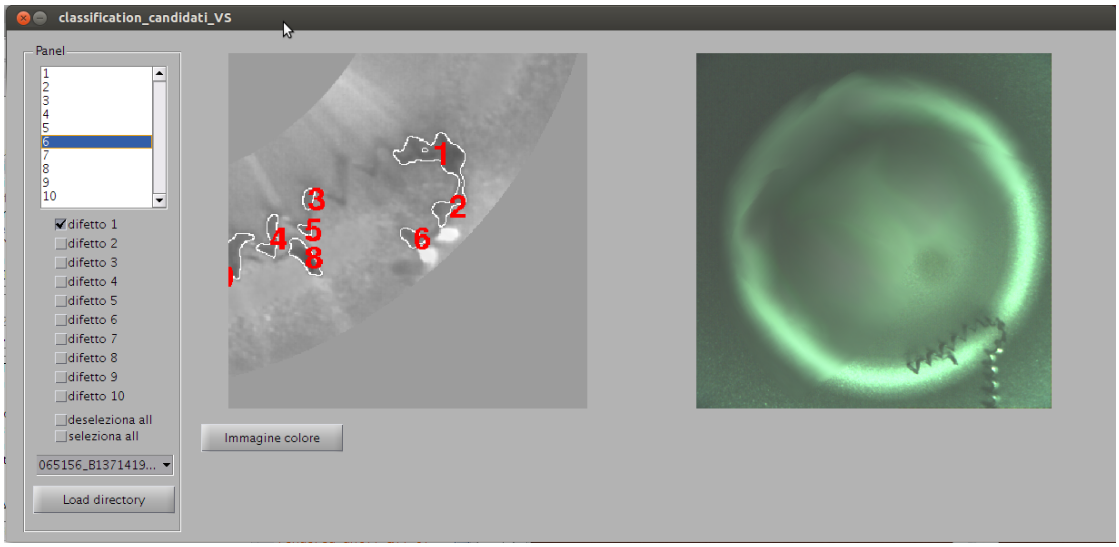


Fig. B.2: Esempio dell'interfaccia GUI classification_candidati_VS utilizzata nella fase di Test

B.3.3 Fase di Test

Nei test esposti nel terzo capitolo, i dati sono stati raccolti automaticamente attraverso funzioni che saranno spiegate nel seguito del paragrafo. Per far ciò è stato necessario andare a classificare manualmente tutti i candidati ottenuti dalle immagini di test. Tale step è stato reso necessario per ottenere i risultati di classificazione, nel caso di nuove prove non risulterebbe necessario.
Come per la fase di training bisogna raccogliere in un'unica cartella, che indicheremo con *<dir_ Test>*, tutte le cartelle ottenute come output dal software di Video Systems, delle immagini che si vogliono classificare. Anche in questo caso sono due gli step da eseguire: determinare e classificare manualmente i candidati ed eseguire i test.

Determinazione dei candidati e classificazione in difetti

Il primo step viene eseguito dallo script *Find_ candidati.m* all'interno del quale bisogna modificare il valore della variabile *directory* con il path di *<dir_ Test>*. Tale funzione esegue i medesimi passi della funzione spiegata nel sotto paragrafo precedente, l'unica variazione risulta sulla struttura salvata nella cartella *CATALOGAZIONE_ VS*.
Per raccogliere in maniera automatica i risultati dei classificatori si è reso necessario andare a classificare i candidati in difetti. Tale step viene svolto attraverso la GUI *classification_ candidati_ VS*, in Figura B.2 è riportato un esempio. L'interfaccia è simile alla precedente, cambiano il numero delle check box. Infatti in tale catalogazione si classificano i difetti e non i candidati. Con ciò si intende, come spiegato nel capitolo 3, che nel caso in cui più candidati identifichino uno stesso difetto come ad esempio i candidati di Figura B.2, andranno classificati selezionando il medesimo check box, ad esempio difetto 1.
La classificazione non è necessaria si è liberi di passare allo step successivo, naturalmente non sarà possibile ottenere i risultati numerici della classificazione ma solo le immagini.

Fase di Test

La fase di classificazioni delle immagini è svolta dallo script *classificazione_ VIDEOSYSTEM.m*. Tale script classifica le immagini, raccoglie i dati statistici nel file *statistiche.csv*, stampa le immagini dei risultati della classificazione e dei falsi positivi e falsi negativi ottenuti. In tale script è necessario settare alcune variabili:

- *directory*, ossia il path di *<dir_ Test>*;

- *directory_ write*, è il path della cartella in cui si vuole salvare le immagini del risultato della classificazione, dei falsi positivi e/o dei falsi negativi;
- *stampa_ immagini*, variabile booleana (0,1) con cui si decide se si vogliono salvare (1), o meno (0), le immagini della classificazione nella cartella *directory_ write*;
- *stampa_ immagini_ FP*, variabile booleana (0,1) con cui si decide se si vogliono salvare (1), o meno (0), le immagini dei falsi positivi ottenuti durante le classificazioni nella cartella *directory_ write/FP*;
- *stampa_ immagini_ FN*, variabile booleana (0,1) con cui si decide se si vogliono salvare (1), o meno (0), le immagini dei falsi negativi ottenuti durante le classificazioni nella cartella *directory_ write/FN*;

Lo script *classificazione_ VIDEOSYSTEM.m* utilizza la funzione *Classification_ CANDIDATI* la quale in ingresso vuole:

- directory, cioè *<dir_ Test>*;
- *name_ dir* è il nome della cartella ottenuta come output dal software videosystem dell'immagine che si vuole catalogare;
- *name_ root* è il path della cartella di cui si vuole analizzare l'immagine;
- *directory_ write*
- *stampa_ immagini*
- *stampa_ immagini_ FP*
- *stampa_ immagini_ FN*
- *K_ lunghi* è il tipo di kernel utilizzato per allenare il classificatore dei difetti lunghi
- *K_ circolari* è il tipo di kernel utilizzato per allenare il classificatore dei difetti circolari
- *K_ altro* è il tipo di kernel utilizzato per allenare il classificatore dei difetti altri

Questa funzione utilizza a sua volta due script:

- *Raccolta_ statistiche_ catalogazione.m*, tramite il quale si va a salvare all'interno del file *risultato.mat*, nella cartella dell'immagine che si sta classificando, tutti i dati della classificazione che riguardano tale immagine;
- *STAMPA_ RISULTATI_ VIDEOSYSTEM.m*, che stampa i risultati della classificazione. Ogni candidato sarà evidenziato nell'immagine originale con un cerchio di colore:
  - giallo, se il candidato è stato classificato come non difetto;
  - rosso, il candidato appartiene all'insieme lunghi ed è stato classificato come difetto;
  - blu, il candidato appartiene all'insieme circolari ed è stato classificato come difetto;
  - verde, il candidato appartiene all'insieme altro ed è stato classificato come difetto;

Per eseguire i test tramite il secondo schema di classificazione bisognerà utilizzare la funzione *classificazione_ VIDEOSYSTEM_ 2classi.m*. Tale funzione ha gli stessi ingressi a meno dell'input *K_ circolari*. Gli script utilizzati da questa funzione per raccogliere i dati di classificazione e stampare le immagini risultano rispettivamente *Raccolta_ statistiche_ catalogazione_ 2classi.m* e *STAMPA_ RISULTATI_ VIDEOSYSTEM_ 2classi.m*. Quest'ultimo script utilizzerà solamente i colori rosso se il candidato è stato classificato come difetto, giallo altrimenti.

All'interno del CD fornito i file relativi a questo problema sono contenuti nella cartella *Primo_ Problema*. In tale cartella i file riguardanti la fase di training sono contenuti nella carella *TRAIN* mentre, quelli della fase di Test sono contenuti nella cartella *TEST*.

B.4 Codice Secondo Problema

Il codice scritto è più snello e meno articolato del primo. Divideremo il codice per la fase di training e per la fase di test che sono salvati all'interno del CD rispettivamente nella cartella *Train* e *Test* all'interno della directory *secondo_problema*. Nel paragrafo successive verranno esposte le funzioni usate sia nella fase di train che in quella di test.

B.4.1 Funzioni generali

Sia nella parte di training e di test è necessario determinare la sezione circolare dell'immagine che rappresenta l'oggetto da analizzare. Al fine di determinare tale parte dell'immagine sono state utilizzate alcune funzioni implementate da noi o scaricate dal sito :

`http://www.mathworks.com/matlabcentral/fileexchange/.`

Tali funzioni sono:

transImageInvPolar che applica una trasformazione polare inversa ad un immagine con struttura circolare. Come output restituisce l'immagine linearizzata, in cui le colonne dell'immagine rappresentano i raggi della sezione circolare di partenza.

motorcase_circlefit tramite tale funzione ricaviamo il centro e il raggio del cerchio della sezione circolare dell'immagine contenente l'oggetto. Per far ciò ci si ricava un immagine binarizzata attraverso una sogliatura del colore. Tramite tale blob si ricava il contorno che sarà passata come ingresso alla funzione **Landau_Smith** che determina la circonferenza che interpola meglio i punti del contorno.

B.4.2 Fase di Training

Per tale fase si è implementata la GUI *Train_features*, la cui interfaccia è riportata in Figura B.3. Prima di eseguire il file è necessario impostare alcune variabili:

- *handles.nameDir* è il path della cartella contenente le immagini che si vogliono utilizzare per il training del classificatore;
- *handles.Kresize* è il fattore di scalatura dell'immagine di partenza. Come detto nel capitolo 4, l'immagine di partenza ha dimensioni eccessivamente grandi. É stato previsto quindi di scalare l'immagini prima di applicare i filtri;
- *handles.cellSize* è il numero di pixel della cella dell'Histogram of Oriented Gradients (HOG)
- *handles.numCell* è il numero di celle quadrate di un lato della finestra che si utilizzerà per scorrere l'HOG, sliding windows

Come si può vedere da Figura B.3 l'interfaccia prevede:

- una list box dove sono elencati tutti i nomi delle immagini contenute in *handles.nameDir*;
- una prima axes, a destra della list box, dove viene mostrata l'immagine selezionata dall'elenco disponibile;
- il pulsante *HOG TRAINING*. Una volta selezionata l'immagine da cui trarre i training examples, premendo tale pulsante si andrà a calcolare l'HOG dell'apertura polare della sezione circolare dell'immagine contenente l'oggetto da esaminare. Tale immagine è riportata nell'axes superiore. Tali passi sono svolti dalla funzione *calculatehog* che viene invoca premendo il pulsante. La funzione richiede in ingresso:

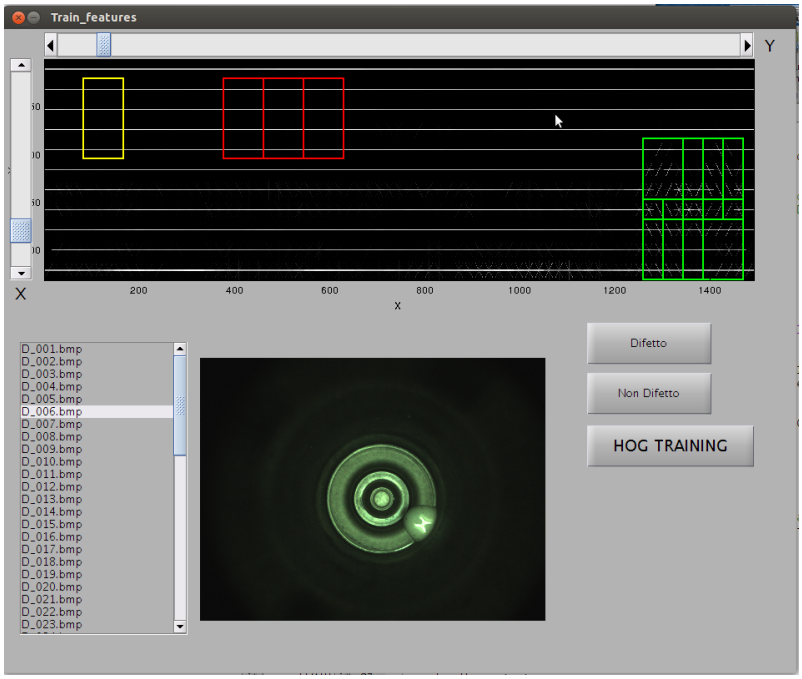


Fig. B.3: Esempio dell'interfaccia GUI *Train_features*, utilizzata nella fase di Train del secondo problema

1. *name_image* il nome dell'immagine;
2. *Kresize*
3. *cellSize*

e come output restituisce l'HOG e l'immagine che lo rappresenta.

- le due slider bar permettono di far scorrere orizzontalmente e verticalmente, la finestra (in giallo) all'interno dell'HOG. Tale finestra permetterà di selezionare i training examples.
- i pulsanti *Difetto* e *Non Difetto* permettono una volta portata la finestra (gialla) nella parte di HOG che vogliamo selezionare come training example, di applicargli la labels difetto o non difetto. Una volta selezionato, il training example sarà colorato in rosso se appartenente alla classe difetto, in verde se appartenente alla classe non difetto. I training examples sono salvati nel file *trainingft.mat* all'interno di un array di strutture. La lunghezza di tale array è pari al numero di immagini da cui sono stati estratti i training examples. La struttura è composta dagli attributi:
  - *pos* e *neg* le coordinate delle celle dell'HOG rappresentanti l'angolo in alto a destra della finestra utilizzata per estrarre un training example appartenente rispettivamente alla classe difetto e non difetto;
  - *FTpos* e *FTneg* il training example (la concatenazione degli istogrammi delle celle della finestra selezionata);
  - *name* è il nome dell'immagine da cui sono stati estratti i training examples.

Lo script file *training_classificatore.m* legge il file *Train_features* e allena il classificatore tramite i parametri impostati nella funzione *svmtrain*.

B.4.3 Fase di Test

Per la fase di test si è sviluppata una seconda GUI *Test_Image*. Un esempio dell'interfaccia è riportato in Figura B.4. Prima di eseguire la GUI è necessario impostare alcuni parametri:

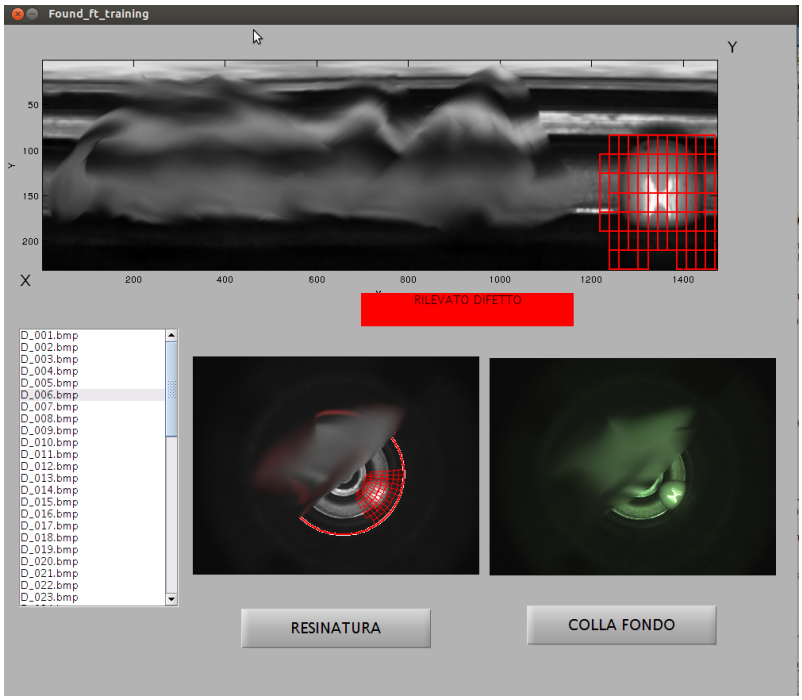


Fig. B.4: Esempio dell'interfaccia GUI *Train_features*, utilizzata nella fase di Test del secondo problema

- *handles.nameDir* è il path della cartella contenente le immagini da analizzare
- *handles.Kresize* è il fattore di scalatura dell'immagine di partenza
- *handles.cellSize* è il numero di pixel della cella dell'Histogram of Oriented Gradients (HOG)
- *handles.numCell* è il numero di celle di un lato della finestra che si utilizzerà per scorrere l'HOG, sliding windows.

Le ultime tre variabili vanno settate in accordo con quelle utilizzate nella fase di training del classificatore. Per eseguire il test è necessario premere il pulsante *COLLA FONDO* e i candidati classificati come difetti verranno segnalati in rosso sia sull'immagine dell'apertura polare dell'oggetto analizzata, la prima immagine in alto, sia sull'immagine di partenza in scala di grigi, prima figura in basso a sinistra. Per andare ad analizzare l'immagine si richiama all'interno della GUI la funzione *MotorCase_GUI* che non fa altro che eseguire ciò che è stato spiegato nel capitolo quattro.

Bibliografia

[1] N. Dalal and B. Triggs, "*Histograms of oriented gradients for human detection*", in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "*Object Detection with Discriminatively Trained Part Based Models*", PAMI, vol. 32, no. 9, pp. 1627-1645, September 2010

[3] Chang, Chih-Chung and Lin, Chih-Jen, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, vol. 2, pp, 27:1–27:27, 2011, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[4] A. Ng, *CS229 Lecture Notes*, Materiale didattico del corso Machine Learning dell'Università di Stanford, avalaible at http://cs229.stanford.edu/materials.html

[5] O. Chapelle, P. Haffner, and Vladimir N. Vapnik, *Support Vector Machines for Histogram-Based Image Classification*, IEEE Transactions on Neural Networks, vol. 10, no. 5, September 1999

[6] M. Stricker and M. Orengo , *Similarity of Color Images*, Communications Technology Laboratory Swiss Federal Institute of Technology, ETH