UNIVERSITA' DELLA CALABRIA

Un Framework di Soluzione ad Alto Livello per Problemi di Classificazione basato su Approcci Metaeuristici

Antonio Candelieri

Tesi di Dottorato
Dottorato di Ricerca in Ricerca Operativa (MAT/09)
XXII° ciclo

Relatore
Prof. D. Conforti

Coordinatore
Prof. L. Grandinetti

Facoltà di Ingegneria
Dipartimento di Elettronica, Informatica e Sistemistica
Laboratorio di Ingegneria della Decisioni per i Servizi Sanitari

Novembre 2009

UNIVERSITY OF CALABRIA

An Hyper-Solution Framework for Classification Problems via Metaheustic Approaches

Antonio Candelieri

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy
XXII° PhD Course in Operations Research (MAT/09)

Supervisor
Prof. D. Conforti

Coordinator
Prof. L. Grandinetti

Faculty of Engineering
Department of Electronics, Informatics and Systems
Laboratory of Decision Engineering for Health Care Delivery

November 2009

La mente è come un paracadute. Funziona solo se si apre

Albert Einstein

La scienza è una cosa meravigliosa... per chi non deve guadagnarsi da vivere con essa

Albert Einstein

UNIVERSITY OF CALABRIA

Abstract

Faculty of Engineering
Department of Electronics, Informatics and Systems
Laboratory of Decision Engineering for Health Care Delivery

Doctor of Philosophy

Antonio Candelieri

This work deals with the development and implementation of a high-level classification framework which combines parameters optimization of a single classifier with classifiers ensemble optimization, through meta-heuristics. Support Vector Machines (SVM) is used for learning while the meta-heuristics adopted and compared are Genetic-Algorithms (GA), Tabu-Search (TS) and Ant Colony Optimization (ACO). Single SVM optimization usually concerns two approaches: searching for optimal parameter values of a SVM with a fixed kernel (Model Selection) or with a linear combination of basic kernels (Multiple Kernel Learning), both approaches have been taken into account. Adopting meta-heuristics avoids to perform time consuming grid-approach for testing several classifier configurations. In particular, starting from canonical formulation of GA, this study proposes some changes in order to take into account specificities of classification learning. Proposed solution has been extensively tested on 8 classification datasets (5 of them are of public domain) providing reliable solutions and showing to be effective. In details, unifying Model Selection, Multiple Kernel Learning and Ensemble Learning on a single framework proved to be a comprehensive and reliable approach, and showing that best solutions have been identified by one of the strategies depending on decision problem and/or available data. Under this respect, the proposed framework may represent a new effective and efficient high-level SVM classification learning strategy.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Il fine ultimo di tutti i problemi di classificazione del mondo reale è trovare un modello decisionale il più affidabile possibile, ovvero in grado di minimizzare la probabilità di errore su nuovi casi. Attualmente esistono due principali strategie o scuole di pensiero: la ricerca della migliore configurazione di una singola tecnica di apprendimento (Model Selection) o la combinazione di più modelli decisionali, differenti per metodologia o configurazione dei parametri (Classifiers Ensemble). Le due strategie sono diametralmente opposte: la prima si affida alla capacità, propria della metodologia di apprendimento, di rappresentare in modo corretto la relazione che lega il valore delle features di ogni istanza alla sua appartenenza ad una classe, mentre la seconda cerca di sfruttare le differenze tra i classificatori di un insieme basale per poter migliorare le prestazioni globali dell'intero sistema di combinazione, il quale può di conseguenza essere considerato un macro-classificatore.
Le esperienze pratiche maturate presso il Laboratorio di Ingegneria delle Decisioni per i Servizi Sanitari hanno spesso dimostrato che non esiste una scelta preferibile in senso generale tra le due strategie, e a lunghe fasi di apprendimento di modelli decisionali è stato spesso necessario far seguire attività rivolte alla combinazione di un sottoinsieme dei classificatori addestrati, al fine di ridurre il rischio di errore combinando gli aspetti positivi dei differenti modelli decisionali.
La definizione di un unico Framework che prendesse contemporaneamente in considerazione le due modalità di approcciare un problema di classificazione, ha costituito il primo e principale obiettivo per il lavoro di ricerca.
Naturalmente, maggiore è il numero di metodologie di apprendimento e/o configurazioni che si vogliono testare in riferimento ad una specifica problematica di classificazione, maggiore sarà l'onere computazionale per la sperimentazione dell'intero Framework; la soluzione proposta prevede l'utilizzo degli approcci evolutivi (nello specifico gli Algoritmi Genetici) per realizzare una ricerca più efficiente della soluzione ottimale, ovvero del modello decisionale più affidabile, a prescindere che esso sia un singolo classificatore o un sistema di combinazione. Il lavoro ha inoltre preso in considerazione la possibilità di realizzare il Framework attraverso altre meta-euristiche di ricerca oltre agli algoritmi

genetici, ovvero Tabu-Search e Ant Colony Optimization.
La soluzione proposta è estremamente generale ed applicabile a qualsiasi metodologia di apprendimento, tuttavia la trattazione si focalizza su una particolare tecnica nota con il termine Support Vector Machine. Tale strategia di apprendimento presenta un'ulteriore problematica: la scelta della funzione kernel più appropriata. Nel momento in cui il kernel viene fissato e si vuole ricercare il miglior valore dei suoi parametri interni si ricade nell'attività di Model Selection, ma esiste anche un ulteriore approccio che consiste nel combinare in modo opportuno alcuni kernel basali per ottenerne uno in grado di sfruttare i diversi vantaggi offerti dai singoli: tale task viene generalmente identificato con il termine di Multiple Kernel Learning. L'implementazione del Framework nel caso della metodologia Support Vector Machines ha permesso di indirizzare anche la problematica di Multiple Kernel Learning, incrementando ulteriormente la capacità di individuare un classificatore (di tipo Support Vector Machine) affidabile.

I risultati della sperimentazione del Framework hanno dimostrato che l'approccio proposto risulta maggiormente efficace, e particolarmente efficiente, rispetto al più classico setting sperimentale, in cui si testano, a tentativi, differenti configurazioni dei parametri di apprendimento, alla ricerca del modello più affidabile (sperimentazione "a griglia"). I modelli decisionali ottimali, individuati su 8 problemi di classificazione binaria, variano tra Multiple Kernel Learning e combinazioni di Support Vector Machines, dimostrando che non esiste una scelta generale o più frequentemente valida, anzi la tipologia di modello più adatto sembra essere dettato dal problema decisionale e/o dai dati a disposizione. Il confronto con le altre meta-euristiche premia comunque l'approccio basato su algoritmi genetici.

La ricerca ha previsto inoltre lo sviluppo di differenti strumenti software, tutti realizzati in Java, in particolare per l'implementazione delle 3 meta-euristiche di ricerca. Al momento della stesura del presente lavoro di tesi è in atto una fase di stabilizzazione di una libreria java relativa agli algoritmi genetici (con le dovute variazioni proposte alla formulazione classica) ed inoltre è stato terminato lo sviluppo di un modulo software (sempre java) che implementa sia il meccanismo di interfaccia con l'utente che l'integrazione tra la libreria relativa agli algoritmi genetici e WEKA; quest'ultimo è uno strumento open-source java che mette a disposizione molte tra le metodologie di apprendimento più attuali, tra cui anche la tecnica di apprendimento Support Vector Machines.

Il lavoro è articolato secondo il seguente schema: nel Capitolo 1 è fornita la definizione del Framework e le principali motivazioni che ne hanno delineato la formulazione, e vengono inoltre illustrati le attuali strategie di combinazione tra classificatori. Il Capitolo 2 riassume le caratteristiche generali della tecnica di apprendimento Support Vector

Machines, le sue formulazioni ed il "kernel trick", illustrando inoltre alcuni lavori di ricerca fondamentali per Model Selection e Multiple Kernel Learning. Il Capitolo 3 offre un'attenta trattazione in merito agli approcci evolutivi, oltre ad illustrare alcuni importanti lavori relativi all'attuale stato dell'arte nell'applicazione di tali approcci per task di Model Selection, combinazione di classificatori e Multiple Kernel Learning, su Support Vector Machines. Nel Capitolo 4 è illustrato l'approccio proposto e le variazioni alla formulazione classica degli algoritmi genetici, e sono inoltre forniti i dettagli implementativi dei vari strumenti software realizzati. Il Capitolo 5 offre una sintesi sulle altre due meta-euristiche di ricerca (Tabu-search e Ant Colony Optimization), utilizzate come parametro di confronto nell'implementazione del Framework. Nel Capitolo 6 sono infine illustrati i risultati della sperimentazione, mentre le sezioni "Conclusioni" e "Sviluppi Futuri" concludono il lavoro.

Capitolo 1

Un Framework di Classificazione ad alto livello

1.1 Definire un Framework di Classificazione ad alto livello

In relazione ai problemi del mondo reale, implementare un task di classificazione richiede spesso la risoluzione di ulteriori e non meno rilevanti problematiche, prima tra tutte la scelta delle metodologie da utilizzare. Sono in genere numerose quelle applicabili al problema in esame e spesso ne sono addirittura disponibili differenti implementazioni e specializzazioni. E' inoltre difficile scegliere a priori una metodologia piuttosto che un'altra: ogni tecnica utilizza un proprio approccio di apprendimento ed una propria struttura per codificare la conoscenza estratta dai dati; prima delle analisi è generalmente impossibile stabilire quale tecnica di apprendimento si presti in modo più efficace alla risoluzione di uno specifico problema. L'approccio più semplice e naturale consiste nell'utilizzare un insieme di possibili metodologie e/o implementazioni.
Ogni algoritmo di classificazione, per giunta, presenta in genere almeno un parametro da settare prima dell'apprendimento, che ne regola le prestazioni sia in termini di bontà di apprendimento, relativamente all'insieme di addestramento (*fitting* sul *training set*), che in termini di capacità di generalizzare su casi non noti. Anche la scelta di tale parametro risulta in genere di difficile individuazione a monte dell'analisi stessa; l'approccio più comune risulta quello noto come sperimentazione "a griglia", secondo il quale vengono identificate una serie di metodologie/implementazioni applicabili al caso in esame ed una serie di valori plausibili per ognuno dei parametri relativi ad ogni metodologia/implementazione. I risultati scaturiti dalla "batteria" sperimentale vengono spesso riportati proprio in una griglia per una più immediata comparazione e per la successiva selezione dei modelli decisionali risultati maggiormente affidabili.

Quanto finora espresso descrive la classica formulazione di un task di classificazione, essenzialmente rappresentato dall'eseguire differenti algoritmi di apprendimento, valutarne le prestazioni e selezionare il più affidabile. Nelle problematiche del mondo reale tuttavia, il task di classificazione non si esaurisce in ciò, ma presenta almeno un'ulteriore fase relativa ad un'eventuale combinazione di un insieme di differenti modelli decisionali estratti dai dati a disposizione.

Osservazioni empiriche ed applicazioni di Machine Learning hanno infatti confermato nel tempo che un dato algoritmo può risultare migliore di altri in relazione ad uno specifico problema o ad uno specifico insieme di dati, tuttavia individuarlo a priori non è banale. Tutti gli algoritmi di apprendimento cercano di trovare un'ipotesi in un certo spazio $\mathcal{H}$ e, la maggior parte di essi, è in grado di trovarne una corretta nel caso in cui disponga di un sufficiente numero di dati rappresentativi per il fenomeno oggetto di studio.
I sistemi costituiti da differenti modelli decisionali combinati tra loro cercano di sfruttare il differente comportamento locale di ogni modello al fine di aumentare l'affidabilità e l'accuratezza dell'intero sistema, addirittura con la speranza che, se qualche modello dovesse fallire, l'intero sistema sia in grado di correggere l'errore.
La realizzazione di tali sistemi è giustificata dai due più comuni scenari reali, relativi alla disponibilità di differenti approcci di apprendimento automatico o di più insiemi di dati differenti tra loro in termini di variabili (features) o di istanze (ad esempio perchè i dati sono stati raccolti in tempi differenti). In entrambi i casi l'obiettivo dei sistemi di combinazione di classificatori rimane minimizzare i possibili errori di predizione su nuovi casi, combinando le capacità predittive di ogni singolo modello.
I due possibili scenari sono schematizzati in figura 1.1. In stretta analogia con il mondo reale, si può pensare ad una consultazione di differenti esperti di dominio con differenti capacità di analisi dei casi (scenario A) o con differenti esperienze professionali (scenario B).

Nel tempo sono state investigate le ragioni per cui le combinazioni di diversi classificatori offrono prestazioni migliori rispetto ai singoli modelli che le costituiscono. Un esempio per tutti è il lavoro di Tom Dietterich [1]: in presenza di un problema di classificazione binaria e di L ipotesi (modelli decisionali), tutti con probabilità di errore inferiore a 0.5, la risultante combinazione, basata su un semplice voto di maggioranza, avrà un errore più basso di ogni singolo classificatore, tanto più basso quanto più gli errori dei singoli classificatori sono tra loro non correlati. Se tale assunzione di indipendenza tra gli errori dei classificatori basali non è soddisfatta non esiste alcuna certezza che l'errore commesso dal sistema di combinazione sia inferiore a quello di ogni singolo classificatore; e nella pratica tale condizione di indipendenza non è purtroppo generalmente soddisfatta. Ciò è essenzialmente dovuto ad un naturale *trade-off* tra l'accuratezza dei classificatori usati nel sistema di combinazione e l'indipendenza tra gli errori da loro commessi (diversità):

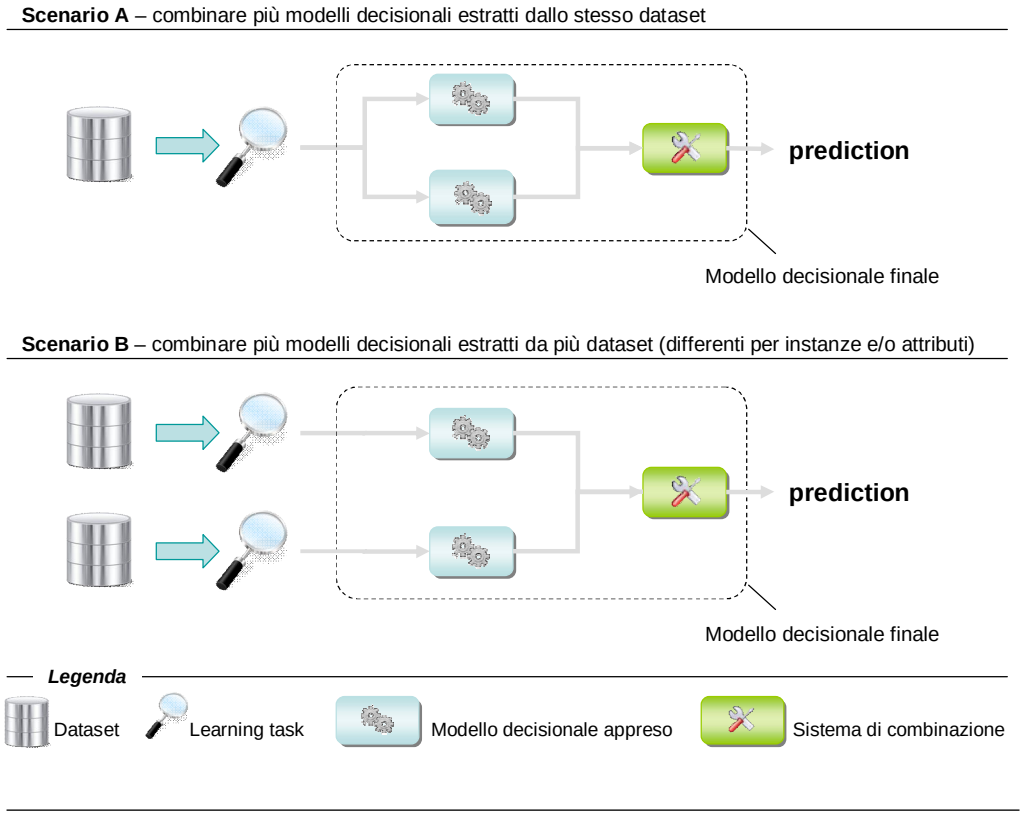

FIGURA 1.1: Possibili scenari di combinazione di modelli decisionali.

più sono accurati i modelli dell'insieme e meno saranno tra loro indipendenti.
Il risultato di Dietterich continua però a suscitare interesse: la probabilità di errore della combinazione di classificatori sarà data dall'area sotto la curva di una distribuzione binomiale con più di $L/2$ ipotesi errate:

$$P_{error} = \sum_{i=L/2}^{L} \binom{L}{i} p^i (1-p)^{L-i}$$

Questo risultato è stato studiato dai matematici fin dalla fine del XVIII secolo nel contesto delle scienze sociali, ed il teorema noto come *Condorcet Jury Theorem* [2] dimostra proprio che il giudizio di un comitato è superiore a quello dei singoli individui "competenti" (probabilità inferiore a 0.5 di essere in errore) che lo compongono. Il teorema non solo giustifica la recente ricerca nel campo della combinazione di differenti modelli decisionali, ma propone anche una linea di ricerca diametralmente opposta all'usuale sviluppo di un singolo classificatore altamente accurato.

Inoltre, specialmente nel dominio medico, in cui l'estrazione di nuova conoscenza o l'identificazione di criteri decisionali/predittivi a supporto delle decisioni degli operatori medici assumono una ancor più forte rilevanza, la richiesta di una spiccata affidabilità

dei modelli decisionali è una prerogativa basilare. Alla luce del *Condorcet Jury Theorem* e del risultato di Dietterich è quindi consigliabile (ma in pratica spesso necessario) combinare le decisioni di differenti modelli in modo da ridurre il rischio di commettere errori.

Esistono differenti tecniche per realizzare la combinazione di classificatori, alcune intuitivamente semplici, come i sistemi di voto (a maggioranza o pesati), ed altre che coinvolgono meccanismi più articolati. Nel successivo paragrafo saranno forniti ulteriori dettagli in merito, mentre al momento risulta fondamentale definire il primo degli obiettivi della presente ricerca, ovvero estendere il concetto di task di classificazione, proponendone una nuova formulazione più ampia e, in un certo senso, più "completa", in grado di includere in modo diretto l'ulteriore passo di combinazione di differenti modelli decisionali.
Va inoltre sottolineato che l'interesse di questo lavoro è stato in particolare rivolto ad una specifica metodologia di classificazione, nota con il nome di Support Vector Machines (SVM). Come ogni altra metodologia di classificazione, anche SVM è interessata dalle problematiche generali fin qui esposte: selezione della migliore configurazione dei parametri ed eventuale combinazione di diversi classificatori (solo SVM o misti), tuttavia essa presenta un'ulteriore problematica, molto simile a quella della selezione della migliore configurazione dei parametri, e comune a tutte le metodologie basate su kernel, appunto l'identificazione della più opportuna funzione kernel. La proposta del framework ad alto livello fornisce una possibile soluzione anche in riferimento a tale tematica.

1.2 Metodi di combinazione

In letteratura esistono molte metodologie relative a schemi per la combinazione di modelli decisionali, raggruppabili ed analizzabili sotto differenti punti di vista. Un'utile classificazione viene fornita da Masulli e Valentini [3], in cui gli autori propongono due macro-categorie: i metodi di combinazione *non-generativi* e *generativi*. Alla prima classe appartengono tutte le metodologie che provano a combinare opportunamente un inseme di modelli decisionali senza generarne attivamente nessun altro. Alla seconda categoria appartengono, di contro, tutti i metodi che generano insiemi di modelli decisionali agendo sui "basali" o sulla struttura dei data set, cercando di migliorare attivamente la diversità e l'accurateza dei singoli classificatori di base.
I metodi di combinazione *non-generativi* sono tutti accomunati dall'utilizzare un insieme predefinito di modelli decisionali ottenuti da una precedente fase di addestramento e che vengono successivamente combinati attraverso un opportuno modulo, in genere dipendente dal tipo di input o di output del problema. Per i task di apprendimento

supervisionato (classificazione e regressione), i metodi *non-generativi* più intuitivi e noti sono i *Sistemi di Voto*. Nella loro versione più semplicistica, ed in particolare per problematiche di classificazione, essi restituiscono la classe più frequentemente predetta dal gruppo di modelli in corrispondenza di ogni singola istanza [4–6]. Una variante più raffinata si ottiene mediante un sistema di voto "pesato" in cui viene assegnato un peso al voto di ogni singolo classificatore dell'insieme dipendentemente dal proprio grado di affidabilità [7]. Ulteriori metodi utilizzano operatori di aggregazione di tipo semplice come Minimo, Massimo, Media, Prodotto e Media Pesata Ordinata [8–10]

I metodi *generativi* possono, di contro, migliorare l'accuratezza totale dell'insieme di modelli decisionali potenziandone direttamente le loro accuratezze e diversità. Per raggiungere tale obiettivo i metodi possono

- modificare la struttura e le caratteristiche dei dati: metodi di *ricampionamento* o di *selezione delle caratteristiche*;
- manipolare l'aggregazione tra classi: metodi di *codifica dell'output*;
- selezionare modelli "specializzati" per una specifica regione dello spazio dei dati: metodi di *miscelazione di esperti* (*mixture of experts*);
- selezionare un sottoinsieme di modelli basali valutandone le prestazioni e le caratteristiche: metodi *test-e-selezione*;
- modificare casualmente gli algoritmi di apprendimento basali: metodi di *casualizzazione*

Le tecniche di ricampionamento (*Resampling methods*) sono utilizzate per generare ipotesi differenti, e tra le più note vanno citate le tecniche di *bootstrapping* e *bagging*. Le prime [11] vengono adoperate per ottenere diversi sottoinsiemi dai dati di addestramento, sottoinsiemi sui quali verrà successivamente testato un algoritmo di apprendimento al fine di ottenere tante ipotesi quanti sono i sottoinsiemi di dati creati. Tali metodologie possono essere particolarmente efficaci con algoritmi di apprendimento sensibili a piccole variazioni del training set, come le reti neurali e gli alberi decisionali. Le tecniche di *bagging* [12] consistono invece nel costruire, attraverso una procedura di bootstrap, repliche dell'insieme di addestramento e successivamente le ipotesi sono combinate per ottenere un unico meccanismo aggregato di predizione, ad esempio attraverso un sistema di voto semplice o pesato [13, 14]. La differenza sostanziale tra le due tecniche risiede nel fatto che nel *bagging* viene utilizzata una distribuzione di probabilità uniforme per costruire i vari "sottoinsiemi" su cui viene successivamente invocato l'algoritmo di apprendimento, mentre nel *bootstrapping* tale algoritmo viene eseguito ad ogni iterazione, utilizzando un

differente peso delle istanze di addestramento tale per cui le istanze più frequentemente classificate in modo scorretto abbiano un peso maggiore; ciò permette alla tecnica di apprendimento di concentrarsi maggiormente su tali istanze riducendone il rischio di errore di classificazione. [10, 15–26]. Altre varianti prevedono che la costruzione dei "sottoinsiemi" di addestramento riguardi insiemi disgiunti (*cross-validated commitees* [27, 28]) o ottenuti attraverso un campionamento senza reinserimento [29].

I metodi di selezione delle caratteristiche (*Features Selection*) consistono nel ridurre il numero di attributi per le istanze, input dei classificatori basali del sistema di combinazione. Il metodo a sottospazio casuale (*Random Subspace Method*) [30, 31] prevede la selezione casuale di un sottoinsieme di caratteristiche ed il successivo assegnamento ad un algoritmo arbitrario. I classificatori costruiti nei sottospazi selezionati sono infine combinati tra loro, generalmente attraverso un meccanismo di voto pesato sulla base dell'accuratezza di ogni singolo modello decisionale. E' stato dimostrato che questo approccio risulta efficace con algoritmi che presentano una curva di apprendimento decrescente ed estratti da insiemi di addestramento critici e con un esiguo numero di istanze [32].
Un ulteriore approccio è noto con il termine di *Input Decimation*. La strategia adottata è ridurre la correlazione tra gli errori dei classificatori del sistema di combinazione, dividendoli attraverso un apprendimento su differenti sottoinsiemi di caratteristiche. La principale differenza con l'approccio precedente è che in Input Decimation le caratteristiche vengono selezionate non casualmente ma in relazione alla correlazione tra il valore di ogni caratteristica e la classe.
Un'altra tecnica particolarmente semplice consiste nel partizionare l'intero insieme delle caratteristiche ed assegnare ogni sottoinsieme ad un classificatore basale del sistema di combinazione [8, 33, 34].

L'idea alla base dei metodi di combinazione noti come *mixtures of experts* è che la combinazione dei classificatori basali venga regolata da un ulteriore algoritmo di apprendimento, operante come "supervisore", ed in grado di selezionare il più opportuno modello decisionale dell'insieme sulla base di ogni istanza di input [35, 36]. Una variante di questo approccio è nota come *hierarchical mixture of experts*, in cui l'output è ottenuto come combinazione non-lineare dei singoli output dei classificatori dell'insieme ed in cui la combinazione è realizzata da più "supervisori" organizzati in un'opportuna struttura gerarchica [36–38].

I metodi di decomposizione di codifica dell'output (*Output Coding*) sono in genere utilizzati in relazione a problemi di classificazione multi-classe: suddividono l'intero problema in sotto-problemi di classificazione binaria e successivamente ricompongono il valore di classe predetto in relazione al problema originale [1, 39, 40].

Nel tempo sono stati proposti differenti approcci di *Output Coding* a seconda di differenti metodi di decomposizione:

- **One-Per-Class (OPC)**. Ogni classificatore binario esegue la separazione di una singola classe C da tutte le altre. [41]
- **Pair Wise Coupling (PWC)**. Ogni classificatore binario f_i esegue la separazione tra 2 classi C_{i1} e C_{i2}, ignorando tutte le altre [42].
- **Correcting Classifiers (CC)** e **Pair Wise Coupling Correcting Classifiers (PWC-CC)**. Varianti del metodo di decomposizione PWC in grado di ridurre l'errore riscontrabile nello schema originale e dovuto al processamento di informazioni non pertinenti da parte dei classificatori binari [43].

Per quanto concerne i metodi di test-e-selezione (*Test-and-Selection*) la tecnica più semplice è di tipo *greedy*: un modello decisionale è aggiunto all'insieme se l'errore del sistema viene conseguentemente ridotto [6], tuttavia, in linea di principio, può essere applicata qualsiasi tecnica di ottimizzazione, come ad esempio gli algoritmi genetici [44] e la Tabu-Search [45]. Possono essere utilizzati anche metodi mutuati dalla *features selection* (ricerca *forward* e *backward*). Infine esistono anche metodi di selezione dinamica dei modelli di un insieme (*Dynamic Classifier Selection*) [46–48] che valutano, ad esempio, l'accuratezza di ogni singolo classificatore dell'insieme in una regione locale dello spazio delle features, precisamente nell'intorno di ogni singola istanza, e selezionano successivamente il modello più affidabile dell'insieme per quella particolare istanza.

Infine, diversi risultati sperimentali hanno dimostrato che sfruttare la casualità per creare i classificatori basali di un insieme migliora le prestazioni di ogni singolo classificatore non-casualizzato. Ad esempio, insiemi di alberi decisionali casualizzati risultano migliori di singoli alberi C4.5 [49], ottenendo addirittura ulteriori margini di miglioramento per l'accuratezza se si aggiungono: rumore bianco ai dati di input, bootstrap e regolarizzazione dei pesi [50].

Capitolo 2

Support Vector Machines

In questo capitolo verranno illustrati i concetti alla base della classificazione a vettori di supporto (Support Vector Machines).

2.1 Formulazioni del problema di classificazione

Nei termini più generali, l'apprendimento supervisionato è definito come il problema di costruire, a partire da un insieme di coppie $Z = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell))$, una funzione decisionale in grado di assegnare ad un qualsiasi vettore di input $\mathbf{x} \in \mathcal{X}$ un'etichetta $y \in \mathcal{Y}$. L'apprendimento supervisionato si divide poi in due differenti problematiche: la *regressione* (in cui le etichette sono valori continui reali, quindi con $\mathcal{Y} \subseteq \mathbb{R}$) e la *classificazione* (in cui le etichette sono valori nominali predefiniti). Sebbene la metodologia SVM venga utilizzata sia per problematiche di regressione che di classificazione, nel presente lavoro di tesi ci si focalizzerà semplicemente sul secondo aspetto, ed in particolare al caso della classificazione binaria, ovvero $|\mathcal{Y}| = 2$.

Prima di inoltrarsi nella trattazione delle formulazioni del problema di classificazione SVM è doveroso puntualizzare alcuni concetti basilari, primo tra tutti il concetto di *iperpiano*.

Iperpiano
Dato uno spazio vettoriale $\mathcal{H}$, un qualsiasi iperpiano è definito come:

$$\{\mathbf{x} \in \mathcal{H} : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}, \ \mathbf{w} \in \mathcal{H}, b \in \mathbb{R} \tag{2.1}$$

In questa formulazione, tuttavia, l'iperpiano gode di una proprietà "superflua": moltiplicando sia $\mathbf{w}$ che b per una stessa costante non nulla si ottiene lo stesso iperpiano

rappresentato in termini di differenti parametri. Tale proprietà viene annullata dalla seguente definizione di *iperpiano canonico*.

Iperpiano Canonico

La coppia $(\mathbf{w}, b) \in \mathcal{H} \times \mathbb{R}$ è detta forma canonica dell'iperpiano (2.1), rispetto ai punti $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{H}$, se è scalato in modo tale che risulti:

$$\min_{i=1\ldots m} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1 \tag{2.2}$$

Ciò equivale a dire che il punto più vicino all'iperpiano ha distanza $1/\|\mathbf{w}\|^2$.
La definizione di iperpiano canonico individua tuttavia una coppia di iperpiani: data la coppia $(\mathbf{w}, b)$, anche la coppia $(-\mathbf{w}, -b)$ soddisfa la 2.2.
Ai fini della classificazione, un iperpiano può essere utilizzato come funzione decisionale:

$$\begin{aligned} f_{\mathbf{w},b} &: \mathcal{H} \to \pm 1 \\ \mathbf{x} &\mapsto f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) \end{aligned} \tag{2.3}$$

I due iperpiani, essendo orientati in modo differente, costituiscono due diverse funzioni decisionali: l'una l'opposta dell'altra. L'obiettivo diviene quindi trovare l'iperpiano (funzione decisionale $f_{\mathbf{w},b} = \mathbf{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ che soddisfi $f_{\mathbf{w},b}(\mathbf{x}_i) = y_i$ per tutti gli elementi dell'insieme Z (caso *linearmente separabile*) o, almeno, per la maggior parte di essi (caso *non-linearmente separabile*).
Se esiste un iperpiano in grado di etichettare correttamente tutti gli elementi dell'insieme Z, la 2.2 implica che:

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 \; \forall i = 1 \ldots m \tag{2.4}$$

Mentre le coppie $(\mathbf{w}, b)$ e $(-\mathbf{w}, -b)$ soddisfano entrambe la forma canonica 2.2, solo una risolve la 2.4, permettendo di distinguere l'orientamento dei due differenti iperpiani.

Margine

Altro concetto di cruciale importanza per un algoritmo di apprendimento SVM è il *margine*. Per un iperpiano $\{\mathbf{x} \in \mathcal{H} : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$, il *margine geometrico* del punto $(\mathbf{x}, y) \in \mathcal{H} \times \{\pm 1\}$ è definito come:

$$\rho_{(\mathbf{w},b)}(\mathbf{x}, y) := y(\langle \mathbf{w}, \mathbf{x} \rangle + b)/\|\mathbf{w}\|$$

Per un punto $(\mathbf{x}, y)$ correttamente classificato il margine corrisponde semplicemente alla distanza del punto dall'iperpiano: si può infatti notare immediatamente che il margine è nullo per un qualsiasi punto che giace sull'iperpiano di separazione. La moltiplicazione per y assicura che il margine sia un valore positivo nel caso in cui il punto sia correttamente classificato (distanza punto-margine), mentre si otterrà un valore negativo per

i punti non correttamente classificati. Infine, nel caso di iperpiani canonici il margine è pari a $1/\|\mathbf{w}\|$, conferendo alla quantità $\mathbf{w}$ un significato geometrico ed un ruolo cruciale per l'algoritmo SVM: ottenere da un insieme di addestramento un iperpiano di separazione con un margine elevato suggerisce infatti una buona affidabilità sulla classificazione di nuovi casi.
Esistono infatti diversi iperpiani di separazione, ma il migliore è appunto quello con margine massimo. Questi aspetti hanno permesso di giungere alla prima formulazione del problema di classificazione SVM, nota come formulazione *hard margin*:

Hard margin SVM

$$\min_{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}} \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.5}$$

$$y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 \;\forall i = 1, \ldots, m \tag{2.6}$$

che è un problema di ottimizzazione in forma primale.
In realtà è molto più conveniente approcciare il problema nella sua forma duale, derivandolo dal primale attraverso la funzione Lagrangiana:

$$L(\mathbf{x}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1) \tag{2.7}$$

con moltiplicatori di Lagrange $\alpha_i \geq 0$ e dove $\alpha = (\alpha_1, \ldots, \alpha_m)$.
La funzione Lagrangiana può essere massimizzata rispetto ai moltiplicatori α_i e minimizzata rispetto a $\mathbf{w}$ e b. Nel punto di sella si avrà:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0$$
$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0$$

che porta a

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.8}$$

e

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{2.9}$$

Il vettore soluzione $\mathbf{w}$ è quindi un'espansione delle istanze di addestramento. Sebbene la soluzione $\mathbf{w}$ sia unica (perchè la funzione obiettivo 2.5 è strettamente convessa ed i vincoli 2.6 sono convessi), non è detto che lo siano i coefficienti α_i.
In accordo con le condizioni Karush-Kuhn-Tucker (KKT), solo i moltiplicatori Lagrangani α_i che non sono nulli per il punto di sella corrispondono a vincoli attivi; formalmente:

$$\alpha_i \left[y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1\right] = 0 \tag{2.10}$$

Tutti gli elementi $\mathbf{x}_i$ tali per cui $\alpha_i > 0$ sono chiamati *Support Vectors* ed in accordo con la 2.10 giacciono esattamente sul margine. Tutte le altre istanze non-support vectors sono irrilevanti per la soluzione: i vincoli 2.6 sono soddisfatti automaticamente e tali istanze non appaiono nell'espansione 2.9, poichè i corrispondenti coefficienti Lagrangiani soddisfano $\alpha_i = 0$.
Sostituendo la 2.8 e la 2.9 nella funzione Lagrangiana 2.7, si giunge alla forma duale del problema di ottimizzazione:

$$\max_{\alpha \in \mathbb{R}} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.11}$$

$$\alpha_i \geq 0, \ i = 1, \ldots, m \tag{2.12}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.13}$$

Sostituendo infine l'espansione 2.9 nella funzione decisionale 2.3 si ottiene un'espressione in termini di prodotti scalari tra l'istanza da classificare e i support vectors:

$$f(\mathbf{x}) = \mathbf{sign} \left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right). \tag{2.14}$$

Esiste infine un ulteriore modo per ottenere la forma duale del problema di ottimizzazione:
Siano C_- e C_+ gli involucri convessi delle istanze appartenenti alle due classi:

$$C_{\pm} := \left\{ \sum_{y_i = \pm 1} c_i \mathbf{x}_i \,\middle|\, \sum_{y_i = \pm 1} c_i = 1, c_i \geq 0 \right\},$$

l'iperpiano con il massimo margine è la secante del più corto segmento congiungente C_+ e C_-. Formalmente ciò può essere scritto come:

$$\min_{c \in \mathbb{R}^m} \left\| \sum_{y_i = 1} c_i \mathbf{x}_i - \sum_{y_i = -1} c_i \mathbf{x}_i \right\|^2$$

$$\sum_{y_i = 1} c_i = 1$$

$$\sum_{y_i = -1} c_i = 1$$

$$c_i \geq 0$$

con il vettore normale $\mathbf{w} = \sum_{y_i=1} c_i \mathbf{x}_i - \sum_{y_i=-1} c_i \mathbf{x}_i$, scalato affinchè sia un iperpiano canonico (2.2). La soglia b è esplicitamente assegnata in modo tale che l'iperpiano intersechi il segmento congiungente i due involucri convessi.

2.2 Il kernel trick e le funzioni kernel

La formulazione fin qui trattata funziona bene nel caso di dati linearmente separabili, tuttavia questa condizione è spesso rara nei problemi reali. Il passo successivo consiste nell'utilizzare delle superfici più generali come funzioni decisionali, introducendo un ulteriore concetto: il *kernel*. In pratica si utilizza una funzione di *mapping* $\Phi : x_i \mapsto \mathbf{x}_i$ in grado di trasformare i dati di input $x_1, \ldots, x_m \in \mathcal{X}$ portandoli in uno spazio a dimensione maggiore in cui eseguire la separazione lineare (si parla generalmente di passaggio da *Input Space* a *Feature Space*).
Considerando ogni $\mathbf{x}$ delle precedenti formulazioni come il risultato della trasformazione $\Phi(x)$, si ottiene che per massimizzare la funzione obiettivo 2.11 e valutare la funzione decisionale 2.14 è necessario calcolare i prodotti scalari $\langle \Phi(x), \Phi(x_i) \rangle$ in uno spazio a dimensione superiore. Questo costoso calcolo è semplificato dall'adozione di una matrice kernel definita positiva tale che:

$$\langle \Phi(x), \Phi(x_i) \rangle = k(x, x_i) \tag{2.15}$$

La funzione decisionale 2.14 diviene quindi:

$$f(x) = \mathbf{sign}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right) \tag{2.16}$$

Per trovare la funzione decisionale 2.14 basta risolvere il seguente problema (cfr. 2.11):

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{2.17}$$

soggetto ai vincoli 2.12 e 2.13.
Numerosi tipi di kernel possono essere utilizzati mantenendo la convessità del problema di programmazione quadratica; ciò è dovuto al fatto che il vincolo 2.13 esclude alcune parti dello spazio dei moltiplicatori Lagrangiani α_i. Se si è in grado di garantire che k sia definita positiva si avrà che $\alpha^T Q \alpha \geq 0$ per tutti i vettori dei coefficienti α che soddisfano il vincolo 2.13, con $Q_{ij} := (y_i y_j k(x_i, x_j))_{ij}$. Inoltre, se la matrice Q è definita positiva il problema convesso può essere risolto in modo efficiente.
Il valore della soglia b può invece essere calcolato prendendo in considerazione le condizioni KKT 2.10: $\alpha_j > 0$ implica (passando per la 2.15) che

$$\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b = y_j$$

quindi la soglia b può ad esempio essere ottenuta come media dei valori

$$b = y_i - \sum_{i=1}^{m} y_i \alpha_i k(x_j, x_i)$$

per tutte le istanze con $\alpha_j > 0$, ovvero per tutti i support vectors.

Esistono differenti tipologie di kernel di base, tra i più noti vanno citati:

- Polinomiale $k(x, x') = \langle x, x' \rangle^d$;
- Gaussiano $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$;
- Sigmoidale $k(x, x') = tanh(\alpha \langle x, x' \rangle + \beta))$;

Soft margin SVM

La formulazione *hard margin*, con o senza kernel, si basa sull'assunzione che esista un iperpiano, lineare o non, in grado di separare le istanze appartenenti alle due diverse classi. Tuttavia, nella pratica, tale iperpiano potrebbe non esistere o potrebbe essere comunque più conveniente permettere che la funzione decisionale possa commettere un certo numero di errori di classificazione: un banale esempio in cui la formulazione *hard margin* fallirebbe è il caso in cui anche una sola istanza sia stata inizialmente etichettata in modo errato, ciò influenzerebbe in modo cruciale la costruzione del margine. Un ulteriore aspetto è legato alla sovra-specializzazione della funzione decisionale sui dati di addestramento (*overfitting*), con conseguente perdita della capacità di classificare correttamente nuove istanze.
Al fine di permettere un certo numero di errori di classificazione è stato proposto il seguente rilassamento sui vincoli di separazione:

$$y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 - \xi_i, \; i = 1, \ldots, m \tag{2.18}$$

con variabili di *slack*

$$\xi_i \geq 0 \; con \; i = 1, \ldots, m \tag{2.19}$$

Affinchè non si ottenga una soluzione banale caratterizzata da elevati valori ξ_i, viene inserito nella funzione obiettivo il termine di penalizzazione $\sum_i \xi_i$. Si giunge così alla formulazione *soft margin* nota come C-SVC (Support Vector Classification con parametro C), in cui un parametro di regolazione $C > 0$ stabilisce il trade-off tra la minimizzazione dell'errore di classificazione e la massimizzazione del margine.

C-SVC

$$\min_{\mathbf{w}\in\mathcal{H},\xi\in\mathbb{R}^m} \tau(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$$

soggetta ai vincoli 2.18 e 2.19.

Una trattazione dettagliata della metodologia SVM e dell'apprendimento basato su kernel è presente in [51].

2.3 Model Selection

In riferimento alla metodologia SVM, un interessante lavoro del 2002 di Chapelle *et al* [52] propone un approccio per il settaggio automatico dei parametri di un classificatore SVM (Model Selection), in particolare del parametro di regolazione C e dei parametri σ del kernel con funzione a base radiale (*Radial Basis Function* - RBF). Va precisato che gli autori utilizzano una definizione differente da quella più usuale per il kernel RBF e che presenta tanti fattori di *scaling* quante sono le dimensioni dell'input:

$$K(\mathbf{x},\mathbf{y}) = exp\left(-\sum_i \frac{(x_i - y_i)^2}{2\sigma_i^2}\right)$$

L'idea alla base dell'approccio proposto è minimizzare una stima dell'errore di generalizzazione attraverso un algoritmo di discesa del gradiente sui parametri della SVM.
Il lavoro si focalizza sulla formulazione *hard margin* ed utilizza, ove necessario, la seguente modifica del kernel (Cortes & Vapnik, 1995 [53]; Cristianini & Shawe-Taylor, 2000 [54]) che permette di considerare la formulazione *soft margin*, con penalizzazione quadratica degli errori, come un caso particolare della formulazione *hard margin*:

$$K = K + \frac{1}{C}I$$

dove I è la matrice identità e C il parametro di regolazione della SVM. Questa modifica permette quindi di considerare C come un ulteriore parametro della funzione kernel.

Gli autori forniscono inoltre una sintesi delle principali tecniche per la stima dell'errore di generalizzazione:

Stima su singola validazione. Quando si hanno sufficienti dati a disposizione, è possibile stimare l'errore di generalizzazione su un insieme di validazione. Tale stima è robusta e la sua varianza diminuisce all'aumentare della dimensione dell'insieme di

validazione. Il valore della stima è dato da:

$$T = \frac{1}{p} \sum_{i=1}^{p} \Psi(-y_i' f(\mathbf{x}_i'))$$

con $(\mathbf{x}_i', y_i')_{1 \leq i \leq p}$ insieme di validazione e Ψ la funzione gradino ($\Psi(x) = 1$ per $x > 0$ e $\Psi(x) = 0$ altrimenti).

Leave-One-Out. Questa procedura consiste nel rimuovere dall'insieme di addestramento un'istanza, costruire il modello decisionale sui rimanenti esempi di addestramento e testarlo successivamente sull'istanza rimossa. Il procedimento viene ripetuto per ognuna delle ℓ istanze dell'inseme di addestramento, costruendo ℓ differenti modelli decisionali. Nonostante tale tecnica rappresenti un buon stimatore dell'errore di generalizzazione risulta essere particolarmente costosa in termini computazionali, visto che l'addestramento deve essere ripetuto per ℓ volte. Sarebbe dunque più opportuno utilizzare un *upper bound* o approssimare lo stimatore con una quantità T più facilmente calcolabile ed avente possibilmente una formulazione analitica.
Denotando con $\mathcal{L}(\mathbf{x}_1, y_1, \ldots \mathbf{x}_n, y_n)$ il numero di errori nella procedura leave-one-out, con f^0 il classificatore ottenuto sull'intero insieme di addestramento e con f^i quello ottenuto quando l'istanza i è stata rimossa, si può scrivere:

$$\mathcal{L}(\mathbf{x}_1, y_1, \ldots \mathbf{x}_n, y_n) = \sum_{p=1}^{n} \Psi(-y_p f^p(\mathbf{x}_p))$$

che può anche essere espressa come:

$$\mathcal{L}(\mathbf{x}_i, y_1, \ldots \mathbf{x}_n, y_n) = \sum_{p=1}^{n} \Psi(-y_p f^0(\mathbf{x}_p) + y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)))$$

Perciò, se U_p è un upper bound per $y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p))$, si otterrà il seguente upper bound per l'errore leave-one-out:

$$\mathcal{L}(\mathbf{x}_i, y_1, \ldots \mathbf{x}_n, y_n) \leq \sum_{p=1}^{n} \Psi(U_p - 1)$$

visto che per la formulazione SVM *hard margin* $y_p f^0(\mathbf{x}_p) \geq 1$ e Ψ è monotonicamente crescente.

Numero di support vectors.
Visto che la rimozione di un non-support vector dall'insieme di addestramento non modifica la soluzione calcolata dall'algoritmo SVM, si può restringere la precedente sommatoria ai soli support vectors e limitare superiormente ogni termine della sommatoria a 1; ciò fornisce il seguente limite sul numero degli errori commessi da una procedura

leave-one-out (Vapnik, 1995 [55]):

$$T = \frac{N_{SV}}{\ell}$$

con N_{SV} il numero di support vectors.

Jaakkola-Haussler bound.
Jaakkola e Haussler [56] (1999) hanno dimostrato la seguente diseguaglianza in riferimento alla procedura leave-one-out per SVM senza soglia ($b = 0$ nella formulazione dell'iperpiano):

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) \leq \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) = U_p$$

che porta al seguente upper bound:

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi(\alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) - 1)$$

Wahba, Lin e Zhang [57](2000) hanno inoltre proposto la seguente stima del numero di errori commessi da una procedura leave-one-out nel caso della formulazione *hard margin*:

$$T = \sum_{p} \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p)$$

che può essere vista come un upper bound della stima proposta da Jaakkola e Haussler, visto che $\Psi(x - 1) \leq x$ per $x \geq 0$.

Opper-Winther bound.
Per la formulazione SVM *hard margin* senza soglia, Opper e Winther [58] (2000) hanno proposto la seguente stima. Sotto l'assunzione che l'insieme di support vectors non muti quando l'esempio p è rimosso dall'insieme di addestramento, si ha:

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \frac{\alpha_p^0}{(\mathbf{K}_{SV}^{-1})_{pp}}$$

dove $\mathbf{K}_{SV}$ è la matrice dei prodotti scalari tra i support vectors. Ciò porta alla seguente stima:

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi \left(\frac{\alpha_p^0}{(\mathbf{K}_{SV}^{-1})_{pp}} - 1 \right)$$

Radius-margin bound.
Per SVMs senza soglia e senza errori di addestramento, Vapnik [59](1998) ha proposto il seguente upper bound sul numero degli errori della procedura leave-one-out:

$$T = \frac{1}{\ell} \frac{R^2}{\gamma^2}$$

dove R e γ sono il raggio ed il margine definiti come:

$$\begin{aligned} \gamma(\mathbf{w}, b, Z) &= \min_{(\mathbf{x}_i, y_i)} \frac{y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b)}{\|\mathbf{w}\|} \\ R(Z) &= \min_{\mathbf{a}, \mathbf{x}_i} \|\Phi(\mathbf{x}_i) + \mathbf{a}\| \end{aligned}$$

Span bound.
Vapnik e Chapelle [60](2000) e Chapelle e Vapnik [61](1999) hanno derivato una stima utilizzando il concetto di *span* dei support vectors. Sotto l'assunzione che l'insieme dei support vectors rimanga lo stesso durante la procedura di leave-one-out, la seguente disuguaglianza risulta soddisfatta:

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \alpha_p^0 S_p^2$$

dove S_p è la distanza tra il punto $\Phi(\mathbf{x}_p)$ e l'insieme Λ_p:

$$\Lambda_p = \left\{ \sum_{i \neq p,\ \alpha_i^0 > 0} \lambda_i \Phi(\mathbf{x}_i), \sum_{i \neq p} \lambda_i = 1 \right\} \tag{2.20}$$

Sotto la precedente assunzione il numero esatto di errori commessi da una procedura di leave-one-out risulta essere

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi(\alpha_p^0 S_p^2 - 1)$$

Se si considera la formulazione SVMs senza soglia, il vincolo $\sum \lambda_i = 1$ può essere rimosso dalla definizione dello *span*. Il valore dello *span* può essere allora limitato superiormente $S_p^2 \leq K((x)_p, \mathbf{x}_p)$, riottenendo il bound proposto da Jaakkola e Haussler.
Per ogni support vector si ha $y_p f^0(\mathbf{x}_p) = 1$. Visto che $\Psi(x-1) \leq x$ per $x \geq 0$, il numero di errori commessi dalla procedura di leave-one-out è limitata da:

$$\sum_p \alpha_p^0 S_p^2$$

E' stato dimostrato (Vapnik e Chapelle, 2000) [60] che lo *span* S_p è limitato dal diametro della più piccola sfera contenente le istanze di addestramento, ed essendo $\sum \alpha_p^0 = 1/\gamma^2$ si ottiene infine:

$$T \leq 4 \frac{R^2}{\gamma^2}$$

strettamente collegato al *radius-margin bound*.
Una derivazione simile a quella utilizzata nello *span bound* è stata proposta da Joachims [62](2000), che limita l'errore della procedura leave-one-out a $\left|\left\{p, 2\alpha_p^0 R^2 > y_p f^0(\mathbf{x}_p)\right\}\right|$, con $0 \leq K(\mathbf{x}_i, \mathbf{x}_i) \leq R^2, \forall i$.

Infine, quando i support vectors non variano, il caso della formulazione *hard margin* senza soglia fornisce lo stesso valore del limite di Opper e Winther:

$$S_p^2 = \frac{1}{(\mathbf{K}_{SV}^{-1})_{pp}}$$

Le stime delle prestazioni di un classificatore SVM, ottenute su un insieme di validazione o attraverso la tecnica di leave-one-out, non sono tuttavia funzioni continue e quindi non sono differenziabili. Affinchè possa essere usato un algoritmo di discesa del gradiente, gli autori propongono l'uso della seguente funzione:

$$\Psi(x) = (1 + exp\,(-Ax + B))^{-1}$$

Tuttavia, le costanti A e B sono di difficile individuazione: se A è troppo piccola la stima non risulterà accurata, mentre se è troppo grande la stima non sarà *smooth*. Piuttosto che tentare di trovare valori 'buoni' per le costanti, essi possono essere ricavati dalla stima delle probabilità a posteriori. Platt ha proposto la seguente distribuzione a posteriori (Platt, 2000) [63]:

$$\tilde{P}_{A,B}(\mathbf{x}) = \tilde{P}(Y = 1|X = \mathbf{x}) = \frac{1}{1 + exp(Af(\mathbf{x}) + B)}$$

dove $f(\mathbf{x})$ è l'output del classificatore SVM. Le costanti A e B possono essere ricavate minimizzando la divergenza di Kullback-Libler tra $\tilde{P}$ ed un'approssimazione empirica di P costruita da un insieme di validazione $(\mathbf{x}'_i, y'_i)_{1 \leq i \leq n_v}$:

$$(A^*, B^*) = \arg\max_{A,B} \sum_{i=1}^{n_v} \left(\frac{1 + y'_i}{2} log(\tilde{P}_{A,B}(\mathbf{x}'_i)) + \frac{1 - y'_i}{2} log(1 - \tilde{P}_{A,B}(\mathbf{x}'_i)) \right)$$

Questa ottimizzazione è eseguita attraverso un algoritmo di discesa del gradiente del secondo ordine (Platt, 2000) [63].
Relativamente a tale stima, la migliore soglia per il classificatore f è tale che $f(\mathbf{x}) = sign(\tilde{P}_{A*,B*}(x) - 0.5)$.

Visto che nell'approccio proposto da Chapelle *et al* il kernel dipende da uno o più parametri, codificabili nel vettore $\theta = (\theta_1, \ldots, \theta_n)$, la funzione del classificatore SVM può essere riscritta nel seguente modo:

$$f_{\alpha,b,\theta(\mathbf{x})} = \mathbf{sign} \left(\sum_{i=1}^{\ell} \alpha_i y_i K_\theta(\mathbf{x}_i, \mathbf{x}_i) + b \right)$$

A questo punto l'obiettivo è scegliere i valori dei parametri α e θ in grado di massimizzare il margine W e minimizzare il criterio di *model selection* T. Più precisamente, fissato θ si vuole trovare $\alpha^0 = \arg\max W(\alpha)$ e scegliere θ^0 tale che:

$$\theta = \arg\min_{\theta} T(\alpha^0, \theta)$$

L'approccio di Chapelle implementa il seguente schema procedurale, in cui l'apprendimento SVM è alternato ad un passo di ottimizzazione nella direzione del gradiente di T nello spazio dei parametri SVM:

1. `Inizializzare` θ `ad un valore casuale`
2. `Usare un algoritmo SVM standard e trovare il massimo di` W`:`

$$\alpha^0(\theta) = \arg\max_{\alpha} W(\alpha, \theta)$$

3. `Aggiornare i parametri` θ `in modo che` T `sia minimizzato (`*`gradient step`*`);`
4. `Se il minimo di` T `è raggiunto STOP altrimenti torna al passo 2.`

Il passo 3 richiede di stimare come vari T al variare di θ; gli autori restringono l'approccio ai casi in cui K_θ sia differenziabile rispetto a θ ed inoltre possa essere calcolato o approssimato il gradiente di T rispetto a θ.
Essendo α^0 dipendente in modo implicito da θ, in quanto definita come massimo di W, la derivata di T^0 rispetto a θ_p assumerà la seguente forma:

$$\frac{\partial T^0}{\partial \theta_p} = \left.\frac{\partial T^0}{\partial \theta_p}\right|_{\alpha^0 \mathbf{fixed}} + \frac{\partial T^0}{\partial \alpha^{\mathbf{0}}} \frac{\partial \alpha^0}{\partial \theta_p}$$

Una volta calcolato il gradiente $\nabla_\theta T(\alpha^0, \theta)$, un modo per eseguire il passo 3 è fare un *gradient step*:

$$\delta\theta_k = -\epsilon \frac{\partial T(\alpha^0, \theta)}{\partial \theta_k}$$

per un ϵ piccolo ed eventualmente decrescente. La convergenza può essere migliorata con l'uso delle derivate del secondo ordine (metodo di Newton):

$$\delta\theta_k = -(\Delta_\theta T)^{-1} \frac{\partial T(\alpha^0, \theta)}{\partial \theta_k}$$

dove l'operatore Laplaciano è definitito da:

$$(\Delta_\theta T)_{i,j} = \frac{\partial^2 T(\alpha^0, \theta)}{\partial \theta_i \partial \theta_j}$$

Nel loro lavoro, Chapelle et al, dimostrano anche come calcolare il gradiente, rispetto ai parametri kernel, per diverse delle stime sintetizzate.

R^2/γ^2 bound.
Per prima cosa viene calcolata la derivata del margine. Nello spazio delle caratteristiche (*features space*), l'iperpiano di separazione ha la seguente espansione:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i^0 y_i \Phi(\mathbf{x}_i)$$

ed è normalizzato in modo tale che

$$\min_{1 \leq i \leq \ell} y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) = 1$$

Dalla definizione di margine risulta $\gamma = 1/\left\|\mathbf{w}\right\|$. Perciò il bound R^2/γ^2 può essere riscritto come $R^2 \left\|\mathbf{w}\right\|^2$.
Grazie al seguente lemma è quindi possibile calcolare la derivata di $\left\|\mathbf{w}\right\|^2$.
Lemma

$$Ep_{err}^{\ell-1} = \frac{1}{\ell} E(\mathcal{L}(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_\ell, y_\ell))$$

dove $p_{err}^{\ell-1}$ è la probabilità dell'errore di test per la SVM addestrata su un campione di dimensione $\ell - 1$ ed i valori attesi sono presi su una scelta casuale del campione.

Può dunque essere dimostrato (Vapnik, 1998) [59] che

$$\frac{1}{2} \left\|\mathbf{w}\right\|^2 = W(\alpha^0)$$

e l'applicazione del lemma precedente al problema di ottimizzazione SVM standard fornisce

$$\frac{\partial \left\|\mathbf{w}\right\|^2}{\theta_p} = -\sum_{i,j=1}^{\ell} \alpha_i^0 \alpha_j^0 y_i y_j \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$$

In secondo luogo va poi calcolata la derivata del raggio della più piccola sfera contenente le istanze di addestramento. Ciò può essere fatto risolvendo il seguente problema di programmazione quadratica (Vapnik, 1998) [59]:

$$R^2 = \max_{\beta} \sum_{i=1}^{\ell} \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$$

soggetto ai vincoli

$$\sum_{i=1}^{\ell} \beta_i = 1$$
$$\forall i \beta_i \geq 0$$

Utilizzando ancora una volta il precedente lemma si ottiene

$$\frac{\partial R^2}{\partial \theta_p} = \sum_{i=1}^{\ell} \beta_i^0 \frac{\partial K(\mathbf{x}_i, \mathbf{x}_i)}{\partial \theta_p} - \sum_{i,j=1}^{\ell} \beta_i \beta j \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$$

dove β^0 massimizza la precedente forma quadratica.

Stima su validazione.
Per quanto riguarda il calcolo del gradiente nel caso di una stima basata sull'errore di validazione, il lavoro di Chapelle et al parte dal calcolo della derivata di α^0 rispetto ai parametri θ del kernel. Per tale calcolo utilizza una formulazione analitica di α^0, partendo dall'assunzione che tutte le istanze che non sono support vectors possono essere rimosse dall'insieme di addestramento senza modificare in alcun modo la soluzione SVM.
Il fatto che tutte le istanze siano support vectors e che giacciano quindi sull'iperpiano di separazione può essere scritto nel seguente modo:

$$\underbrace{\begin{pmatrix} \mathbf{K}^Y & \mathbf{Y} \\ \mathbf{Y}^T & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \alpha^0 \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}$$

dove $\mathbf{K}^Y_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. Se n sono i support vectors, la matrice $\mathbf{H}$ ha dimensioni $(n+1) \times (n+1)$.
I parametri SVM possono essere scritti come:

$$(\alpha^0, b)^T = \mathbf{H}^{-1}(1 \ldots 1\ 0)^T$$

A questo punto è possibile calcolare la derivata di tali parametri rispetto ad un parametro θ_p del kernel. Inoltre, visto che la derivata di una matrice inversa M rispetto ad un parametro θ_p può essere scritta come:

$$\frac{\partial \mathbf{M}^{-1}}{\partial \theta_p} = -\mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \theta_p} \mathbf{M}^{-1} \tag{2.21}$$

segue che

$$\frac{\partial (\alpha^0, b)}{\theta_p} = -\mathbf{H} \frac{\partial \mathbf{H}}{\partial \theta_p} \mathbf{H}^{-1}(1 \ldots 1\ 0)^T$$

ed infine

$$\frac{\partial(\alpha^0, b)}{\partial\theta_p} = -\mathbf{H}^{-1}\frac{\partial\mathbf{H}}{\partial\theta_p}(\alpha^0, b)^T$$

Da questo calcolo si può poi ottenere $\frac{\partial\|\mathbf{w}\|^2}{\partial\theta_p}$.
Sia $\tilde{\alpha} = (\alpha^0, b)$, si avrà $\|\mathbf{w}\|^2 = (\alpha^0)^T\mathbf{K}^Y\alpha^0 = \tilde{\alpha}^T\mathbf{H}\tilde{\alpha}$. Risulterà dunque

$$\begin{aligned}
\frac{\partial\,\|\mathbf{w}\|^2}{\partial\theta_p} &= \tilde{\alpha}^T\frac{\partial\mathbf{H}}{\partial\theta_p}\tilde{\alpha} + 2\tilde{\alpha}\mathbf{H}\frac{\partial\tilde{\alpha}}{\partial\theta_p} \\
&= \tilde{\alpha}^T\frac{\partial\mathbf{H}}{\partial\theta_p}\tilde{\alpha} - 2\tilde{\alpha}\mathbf{H}\mathbf{H}^{-1}\frac{\partial H}{\partial\theta_p}\tilde{\alpha} \\
&= -\tilde{\alpha}^T\frac{\partial\mathbf{H}}{\partial\theta_p}\tilde{\alpha} \\
&= -(\alpha^0)^T\frac{\partial\mathbf{K}^Y}{\partial\theta_p}\alpha^0
\end{aligned}$$

Span

Ultima stima considerata da Chapelle et al è relativa allo *Span*. Ricordando che lo span di un vettore $\mathbf{x}_p$ è definito come la distanza tra $\Phi(\mathbf{x}_p)$ e l'insieme Λ_p precedentemente definito in 2.20, il valore dello span può essere scritto come:

$$S_p^2 = \min_{\lambda}\max_{\mu}\left(\Phi(\mathbf{x}_p) - \sum_{i\neq p}\lambda_i\Phi(\mathbf{x}_i)\right)^2 + 2\mu\left(\sum_{i\neq p}\lambda_i - 1\right)$$

dove i moltiplicatori Lagrangiani μ sono stati inseriti per rafforzare il vincolo $\sum_{\lambda_i} = 1$. Introducendo il vettore esteso $\tilde{\lambda} = (\lambda^T\mu)^T$ e la matrice estesa dei prodotti scalari tra i support vectors

$$\tilde{\mathbf{K}}_{\mathbf{SV}} = \begin{pmatrix} \mathbf{K} & \mathbf{1} \\ 1^T & 0 \end{pmatrix}$$

il valore dello span può essere scritto come:

$$S_p^2 = \min_{\lambda}\max_{\mu}(K(\mathbf{x}_p, \mathbf{x}_p) - 2\mathbf{v}^T\tilde{\lambda} + \tilde{\lambda}^T\mathbf{H}\tilde{\lambda})$$

dove $\mathbf{H}$ è la sottomatrice di $\tilde{\mathbf{K}}_{SV}$ con riga e colonna p rimosse e $\mathbf{v}$ è la colonna p-esima di $\tilde{\mathbf{K}}_{SV}$. Considerando che il valore ottimale di $\tilde{\lambda}$ è $\mathbf{H}^{-1}\mathbf{v}$ segue che

$$S_p^2 = K(\mathbf{x}_p, \mathbf{x}_p) - \mathbf{v}^T\mathbf{H}^{-1}\mathbf{v} = 1/(\tilde{\mathbf{K}}_{SV}^{-1})_{pp} \tag{2.22}$$

L'ultima uguaglianza deriva dalla seguente relazione nota come formula di 'Woodbury' (Lutkepohl, 1996) [64]

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{A}^T \\ \mathbf{A} & \mathbf{A}_2 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}^T \\ \mathbf{B} & \mathbf{B}_2 \end{pmatrix}$$

in cui $\mathbf{B}_1 = (\mathbf{A}_1 - \mathbf{A}\mathbf{A}_2^{-1}\mathbf{A}^T)^{-1}$

Combinando l'equazione 2.22 con la 2.21, la derivata dello *span* risulta:

$$\frac{\partial S_p^2}{\partial \theta_p} = S_p^4 \left(\tilde{\mathbf{K}}_{SV}^{-1} \frac{\partial \tilde{\mathbf{K}}_{SV}}{\theta_p} \tilde{\mathbf{K}}_{SV}^{-1} \right)_{pp}$$

La complessità nel calcolare la derivata dello *span* rispetto ad un parametro θ_p del kernel richiede il solo calcolo di $\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$ e l'inversione della matrice $\tilde{\mathbf{K}}_{SV}$. La complessità di tali operazioni non è maggiore di quella dell'ottimizzazione quadratica del problema stesso.
Tuttavia tale approccio presenta un problema: il valore fornito dalla regola *span* non è continuo; a variazioni *smooth* dei parametri θ i coefficienti α_p variano in modo continuo, ma non S_p^2. Ciò può facilmente essere compreso osservando l'equazione 2.20: modificando il valore dei parametri da θ in $\theta + \epsilon$, un'istanza $\mathbf{x}_m$ potrebbe non essere più un support vector, allora per tutti gli altri support vectors $(\mathbf{x}_p)_{p \neq m}$ l'insieme Λ_p diventerà più piccolo ed una una discontinuità potrebbe probabilmente apparire per il valore $S_p = d(\Phi(\mathbf{x}_p), \Lambda_p)$.
Per risolvere il problema gli autori suggeriscono, come migliore soluzione, l'inserimento di un termine di regolarizzazione nella formula dello *span*:

$$S_p^2 = \min_{\lambda, \sum \lambda_i = 1} \left\| \Phi(\mathbf{x}_p) - \sum_{i \neq p}^{n} \lambda_i \Phi(\mathbf{x}_i) \right\|^2 + \eta \sum_{i \neq p}^{n} \frac{1}{\alpha_i^0} \lambda_i^2$$

Con questa nuova definizione dello *span* l'equazione 2.22 diviene:

$$S_p^2 = 1/(\tilde{\mathbf{K}}_{SV} + \mathbf{D})_{pp}^{-1} - \mathbf{D}_{pp}$$

dove $\mathbf{D}$ è una matrice diagonale con elementi $\mathbf{D}_{ii} = \eta/\alpha_i^0$ e $\mathbf{D}_{n+1,n+1} = 0$. Gli autori nel loro lavoro pongono $\eta = 0.1$, ottenendo una funzione *span* più *smooth* senza modificare il valore del minimo; inoltre il calcolo della derivata per questa nuova espressione non è più complesso rispetto alla precedente.

I benefici che gli autori hanno riscontrato dal loro approccio sono stati molteplici. Primo fra tutti la possibilità di ottimizzare un gran numero di parametri utilizzando la formulazione del Kernel RBF con tanti fattori di *scaling* quante sono le caratteristiche di input. Le loro sperimentazioni hanno inoltre dimostrato che non è necessario ricorrere a stime accurate dell'errore di generalizzazione, ma che al contrario una stima semplice come R^2/γ^2 si comporta in modo eccellente per la finalità di trovare i valori ottimali; inoltre tale stima offre grossi vantaggi essendo facile da calcolare e derivare. Infine questo approccio permette di evitare di isolare dati per la validazione, usando perciò l'intero

insieme di addestramento per l'ottimizzazione dei parametri, al contrario dei metodi di cross-validazione.
Sfortunatamente i risultati riportati dagli autori riguardano solo le prestazioni dell'approccio in merito all'accuratezza dei classificatori estratti, mentre mancano completamente indicazioni sui tempi di calcolo.

2.4 Multiple Kernel Learning

Le tematiche discusse nelle precedenti sezioni possono essere considerate generali rispetto alle metodologie di classificazione: la maggior parte delle tecniche ha almeno un parametro che ne regola l'apprendimento e di cui si vuole quindi determinare il valore ottimale (Model Selection), e diversi modelli decisionali, appresi dai dati, possono collaborare in un sistema di combinazione per implementare un macro-modello decisionale.
Le SVMs presentano in più un ulteriore aspetto, comune a tutte le metodologie basate su kernel, per l'appunto la scelta del kernel. Il successo dei classificatori SVM può essere infatti attribuito alla congiunzione di una procedura robusta di classificazione (la massimizzazione del margine dell'iperpiano di sepearazione) e di un'opportuna trasformazione dei dati (kernel). L'apprendimento SVM può quindi essere concettualmente diviso in due task: il pre-processamento dei dati e la classificazione lineare; è stato dimostrato che le prestazioni dipendono molto più dal pre-processamento che dalla tecnica di classificazione (per esempio, è stato dimostrato che un *kernel perceptron* ed una SVM, con lo stesso kernel, hanno prestazioni simili).
Le prime soluzioni proposte in merito al problema dell'individuazione del kernel ottimale furono di natura parametrica: scelto un tipo di kernel si vanno ad ottimizzare i suoi parametri. Tra queste tecniche rientra il lavoro di Chapelle presentato nella precedente sezione, ma anche approcci basati su meta-euristiche quali Tabu-Search e Algoritmi Genetici, oggetto di trattazione nei prossimi capitoli.
Più di recente è nata l'idea di utilizzare classi non-parametriche dei kernel, il primo lavoro è stato proposto da Cristianini *et al* [65]. Gli autori lavorano in un setting di *trasduzione*, ovvero su problemi in cui è noto un insieme test di istanze. L'obiettivo è individuare una matrice definita positiva, detta matrice di Gram (il kernel), ottimizzando l'*allineamento* tra il kernel ed i dati.
Per allineamento (empirico) tra due kernel k_1 e k_2, rispetto allo stesso campione S di istanze non etichettate, si intende la quantità

$$A(S, k_1, k_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}}}$$

dove con $\langle ., . \rangle_F$ è rappresentato il prodotto scalare tra matrici di Gram (Frobenius): $\langle K_1, K_2 \rangle_F = traccia(K_1^T, K_2) = \sum_{i,j=1}^{n} k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j)$.
Il criterio è semplice da calcolare e da ottimizzare, tuttavia non presenta alcun legame con l'errore di generalizzazione.
Lanckriet *et al* [66] hanno successivamente derivato un bound dell'errore di generalizzazione per il setting trasduttivo, utilizzandolo per la scelta del kernel. Il criterio è convesso e la parametrizzazione del kernel si basa sulla combinazione lineare di un insieme di matrici kernel date. La risoluzione del problema richiede l'utilizzo della programmazione semidefinita (Semidefinitive Programming - SDP).
L'idea è di ottimizzare l'allineamento tra un insieme di etichette ed una matrice kernel attraverso la programmazione semidefinita, dove il kernel può anche essere una combinazione lineare di kernel fissati.
Nel caso in cui il vettore $\mathbf{y}$ sia noto, $K_2 = \mathbf{yy^T}$ può essere considerato come kernel target, con $k_2(x_i; x_j) = 1$ se $y_i = y_j$ e $k_2(x_i; x_j) = -1$ se $y_i \neq y_j$. La misura di allineamento del kernel k_1 con il kernel k_2 può essere espresso come:

$$A(S, k_1, k_2, yy^t) = \frac{\langle K_1, yy^T \rangle}{\sqrt{\langle K_1, K_1 \rangle \langle yy^T, yy^T \rangle}} = \frac{\langle K_1, yy^T \rangle}{m\sqrt{\langle K_1, K_1 \rangle}}$$

con $\langle yy^T, yy^T \rangle = m^2$.
Si supponga di lavorare in un setting trasduttivo, con $S_{n_{tr}}$ l'insieme di addestramento, etichettato, e T_{n_t} l'insieme di test, non etichettato. La matrice kernel corrispondente assume la forma:

$$K = \begin{pmatrix} \mathbf{K}_{tr} & \mathbf{K}_{tr,t} \\ \mathbf{K}_{tr,t}^T & \mathbf{K} \end{pmatrix}$$

dove $\mathbf{K}_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$, $i, j = 1, \ldots, n_{tr} + n_t$.
L'obiettivo del setting trasduttivo è etichettare le istanze dell'insieme test, ovvero apprendere i blocchi $\mathbf{K}_{tr,t}$ e $\mathbf{K}_t$ ottimali. Lo spazio di ricerca delle possibili matrici kernel dovrebbe essere controllato in modo tale da prevenire *overfitting* raggiungendo un buona generalizzazione sui dati di test. In prima istanza è possibile definire un problema di ottimizzazione generale in cui la matrice kernel $\mathbf{K}$ è ristretta al cono semidefinito positivo (insieme convesso), successivamente gli autori considerano un caso specifico: lo spazio di ricerca delle possibili matrici kernel giace nell'intersezione tra un sottospazio lineare ed il cono semidefinito positivo.

Caso 1

La matrice kernel $\mathbf{K}$ è ristretta al cono semidefinito positivo. Sia $\mathcal{K} = \{\mathbf{K} \succeq 0\}$ l'insieme di tutte le matrici semidefinite positive, la matrice $\mathbf{K} \in \mathcal{K}$ con il massimo allineamento

rispetto al vettore $\mathbf{y} \in R^{n_{tr}}$ è ottenuta risolvendo il seguente problema di ottimizzazione:

$$\begin{aligned} \max_{\mathbf{K}} \quad & \mathbf{A}(\mathbf{S}, \mathbf{K}_1, yy^T) \\ s.t. \quad & \mathbf{K} \in \mathcal{K} \\ & trace(\mathbf{K}) = 1 \end{aligned} \tag{2.23}$$

oppure, in modo equivalente:

$$\begin{aligned} \max_{\mathbf{K}} \quad & \left\langle \mathbf{K}_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ s.t. \quad & trace(\mathbf{K}) = 1 \\ & \mathbf{K} \in \mathcal{K} \\ & \langle \mathbf{K}, \mathbf{K} \rangle_F = 1 \end{aligned} \tag{2.24}$$

Introducendo una matrice $\mathbf{M}$ tale che $\mathbf{K}^T\mathbf{K} \preceq M$ e $trace(\mathbf{M}) \leq 1$, può essere riscritto come:

$$\begin{aligned} \max_{\mathbf{M},\mathbf{K}} \quad & \left\langle \mathbf{K}_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ s.t. \quad & trace(\mathbf{M}) \leq 1 \\ & \begin{pmatrix} \mathbf{M} & \mathbf{K}^T \\ \mathbf{K} & I_n \end{pmatrix} \succeq 0; \\ & \mathbf{K} \in \mathcal{K} \\ & \langle \mathbf{K}, \mathbf{K} \rangle_F = 1 \end{aligned} \tag{2.25}$$

con I_n matrice identità di dimensione n.

Caso 2 K appartiene all'insieme di matrici semidefinite positive con traccia limitata ed esprimibili come combinazione lineare di un insieme di matrici kernel fissate $\mathbf{K}_1, \ldots, \mathbf{K}_g$, ed i coefficienti μ_i sono vincolati ad essere non negativi. In questo caso $\mathcal{K}$ è l'insieme che soddisfa i seguenti vincoli:

$$\begin{aligned} \mathbf{K} &= \sum_{i=1}^{g} \mu_i \mathbf{K}_i \\ \mathbf{K} &\succeq 0 \\ \mu_i \geq 0, \quad & i = 1, \ldots, g \\ trace(\mathbf{K}) &\leq c \end{aligned} \tag{2.26}$$

L'insieme iniziale $\mathbf{K}_1, \ldots, \mathbf{K}_g$ dei kernel candidati può contenere kernel lineari, polinomiali (normalizzati), Gaussiani, con differenti valori dei parametri per ognuno.
Aggiungendo i vincoli 2.27 alla 2.25, e portando il tutto in una forma SDP simile alla

2.26, si ottiene:

$$\begin{aligned} \max_{\mathbf{M},\mu_i} \quad & \left\langle \sum_{i=1}^{g} \mu_i \mathbf{K}_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ s.t. \quad & trace(\mathbf{M}) \leq 1 \\ & \begin{pmatrix} \mathbf{M} & \sum_{i=1}^{g} \mu_i \mathbf{K}_i^T \\ \sum_{i=1}^{g} \mu_i \mathbf{K}_i & I_n \end{pmatrix} \succeq 0; \\ & \sum_{i=1}^{g} \mu_i \mathbf{K}_i \succeq 0 \end{aligned} \tag{2.27}$$

Successivamente, Bousquet e Herrmann [67] hanno proposto una soluzione che combina i vantaggi degli approcci precedenti, proponendo alcune classi di kernel ed utilizzando un semplice, veloce ed efficiente algoritmo di discesa del gradiente invece della programmazione semidefinita.

Molto più recentemente (2009), Conforti e Guido [68] hanno riproposto l'utilizzo della programmazione semidefinita con sostanziali differenze rispetto a quanto presentato da Cristianini e Lancriet. Innanzi tutto, la nuova soluzione avanzata dagli autori prevede l'applicazione a problemi di *induzione*, mentre Cristiani e Lancriet basano il loro studio su problematiche di *trasduzione*. Inoltre, la formulazione di Conforti e Guido introduce un vincolo sui pesi della combinazione lineare di kernel, più precisamente un upper-bound sulla loro somma, eliminando qualsiasi vincolo sulla traccia della matrice kernel. Cristiani e Lancriet, infatti, ricercano un'opportuna matrice che abbia un prefissato valore costante della traccia, tuttavia il "miglior" valore da assegnare è di difficile determinazione. Gli autori riportano interessanti ed incoraggianti risultati ottenuti su diversi dataset pubblici afferenti al dominio medico, evidenziando di conseguenza anche l'importanza della maggiore affidabilità riscontrata dalla loro soluzione nelle difficili attività di diagnosi.

Capitolo 3

L'Approccio Evolutivo

3.1 I concetti di base

L'approccio evolutivo trae ispirazione dalla biologia, dalla genetica e dalla natura in genere, imitando i meccanismi della ben nota teoria della "selezione naturale" inizialmente proposta da Charles Darwin. Secondo tale teoria esiste un processo evolutivo (l'evoluzione naturale appunto) che procede senza alcuno scopo preciso e che è principalmente guidato da due componenti, una completamente casuale e l'altra relativa alla sopravvivenza del "più adatto" all'ambiente. Le tecniche evolutive permettono di risolvere problematiche di ottimizzazione utilizzando una stretta analogia con l'evoluzione delle popolazioni: così come gli individui all'interno di una popolazione differiscono tra loro per capacità di adattamento all'ambiente (in termini biologici *fitness*), le soluzioni ad un problema differiscono tra loro per qualità, ovvero per costo o per merito rispetto alla funzione obiettivo. Ogni soluzione è il corrispettivo di un individuo della popolazione, mentre il problema è l'ambiente in cui la popolazione evolve. Se il processo di evoluzione naturale permette ad una popolazione di individui di adattarsi all'ambiente che la circonda, sarà allo stesso modo in grado di far "evolvere una popolazione di soluzioni" verso quelle migliori ed eventualmente, con il tempo, a quelle ottime.

George E. P. Box, statistico noto per la massima "tutti i modelli sono sbagliati, ma alcuni sono utili", fu il primo a pensare che per risolvere un problema di ottimizzazione potevano essere utilizzati meccanismi simili a quelli del processo evolutivo: la "selezione" del più adatto e la "mutazione" casuale delle soluzioni. Il suo lavoro risale almeno agli anni cinquanta e, senza alcun uso dell'elaboratore elettronico, giunse a formulare una metodologia statistica che sarebbe divenuta di largo uso nell'industria, da lui battezzata *evolutionary operation* [69].
Più o meno negli stessi anni, altri studiosi concepirono l'idea di simulare l'evoluzione

su un elaboratore elettronico, ma va al biomatematico Hans J. Bremermann il merito di avere riconosciuto per primo un processo di ottimizzazione nell'evoluzione biologica [70]. Questi primi sforzi incontrarono tuttavia molto scetticismo, ma fu probabilmente per merito dell'aumento della potenza computazionale degli elaboratori elettronici che si rese finalmente possibile la messa in pratica del calcolo evoluzionistico.
Gli algoritmi evolutivi, nelle loro varianti originarie, furono inventati indipendentemente e praticamente nello stesso periodo, a metà degli anni Sessanta, nel seno di tre distinti gruppi di ricerca: in America, Lawrence Fogel e colleghi, dell'Università della California a San Diego, posero le basi della programmazione evolutiva (*evolutionary programming*) [71], mentre presso l'Università del Michigan, ad Ann Arbor, John Holland proponeva i primi algoritmi genetici (*genetic algorithms*) [72]; in Europa, invece, furono Ingo Rechenberg e colleghi, allora studenti presso il Politecnico di Berlino, a ideare quelle che battezzarono "strategie evolutive" (*Evolutionsstrategien*) [73]. Per i successivi 25 anni i tre filoni si svilupparono essenzialmente in modo indipendente, finché nel 1990 non si assistì ad uno sforzo organizzato per farli convergere: la prima edizione del congresso PPSN (*Parallel Problem Solving from Nature*), tenuto quell'anno a Dortmund. Da allora i ricercatori interessati al calcolo evoluzionistico (*evolutionary computation*) formano un'unica, anche se articolata, comunità scientifica.

3.2 Gli Algoritmi Genetici

Gli Algoritmi Genetici (*Genetic Algorithms*, GA) sono tra gli approcci evolutivi più noti e rappresentano un'efficace meta-euristica per problematiche di ricerca ed ottimizzazione. In realtà non è garantito che un GA fornisca un ottimo globale del problema, ma è generalmente in grado di fornire soluzioni soddisfacenti in tempi più che ragionevoli.
Gli ingredienti base della tecnica sono:

- La *popolazione*, ovvero un insieme di soluzioni possibili per il problema (individui della popolazione)
- Una *funzione di fitness* in grado di calcolare quanto ogni soluzione risolva bene il problema (quanto ogni individuo della popolazione sia adatto all'ambiente)
- Gli *operatori genetici*, ovvero le operazioni che implementano effettivamente la ricerca all'interno dello spazio delle soluzioni (evoluzione degli individui della popolazione). I due operatori genetici di base sono il *cross-over* (o ricombinazione) e la *mutazione*. In termini semplicistici: il cross-over genera nuove soluzioni ricombinandone due (o più) tra quelle presenti nella popolazione corrente e trasmettendo

le caratteristiche dei genitori ai figli, mentre la mutazione effettua una perturbazione di alcune soluzioni in modo da ampliare lo spazio di ricerca tra le soluzioni possibili.

L'ultimo aspetto, relativo agli operatori genetici, è senza dubbio il punto cardine della meta-euristica: il cross-over e la mutazione sono in realtà due meccanismi antagonisti, l'*exploitation* della migliore soluzione corrente e l'*exploration* nello spazio di ricerca, ovvero ricerca *locale* e *globale* rispettivamente. Un bilanciamento tra questi due meccanismi è cruciale: una exploitation eccessiva intrappolerebbe il GA in un ottimo locale non soddisfacente, al contrario una exploration eccessiva non utilizzerebbe in modo efficiente la conoscenza già disponibile (migliori individui nella popolazione corrente), rallentando oltremodo la ricerca.
Per comprendere maggiormente a fondo i dettagli della metodologia GA si farà riferimento alla sua forma canonica. Le soluzioni, ovvero gli individui della popolazione, sono dette *cromosomi* e sono rappresentate con stringhe di bit a lunghezza fissa. Ogni singolo bit della stringa rappresenta un gene, con valore 0 o 1. Il numero di cromosomi all'interno della popolazione è costante durante l'intero processo di evoluzione: ciò significa che i "figli più adatti" rimpiazzeranno i "genitori meno adatti". In figura 3.1 è riportato lo schema di principio di un GA canonico. La popolazione viene inizializzata in modo

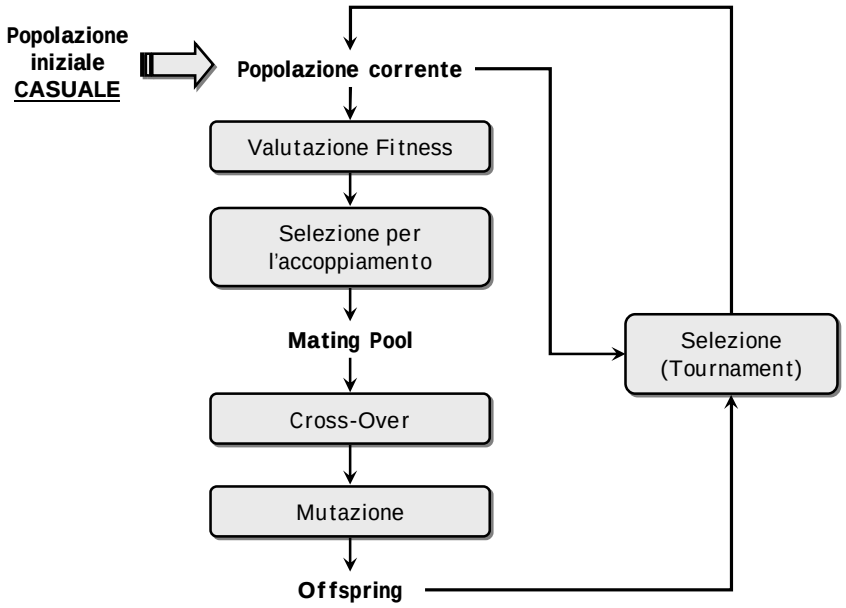

FIGURA 3.1: Schema di principio di un Algoritmo Genetico classico

casuale e successivamente viene valutato il fitness di ogni soluzione. Sulla base dei valori ottenuti, viene eseguita una selezione per l'accoppiamento, in cui un numero di soluzioni

pari alla dimensione della popolazione viene selezionato con probabilità proporzionale al valore di fitness (e con reinserimento); viene così costituito un insieme di individui adatti alla riproduzione ovvero di cui è vantaggioso trasmettere le caratteristiche. Dall'insieme degli individui selezionati per l'accoppiamento (*mating pool*) si scelgono casualmente due genitori e si effettua l'operazione di cross-over in grado di generare due figli. Le operazioni di selezione di una coppia di genitori dal mating pool e di cross-over vengono ripetute fin quando non è stato generato un numero di figli pari alla dimensione della popolazione. E' importante evidenziare che, utilizzando una probabilità proporzionale al fitness per selezionare gli individui adatti all'accoppiamento (ed essendo la selezione con reinserimento), le migliori soluzioni correnti avranno una maggiore probabilità di essere ripetute all'interno del mating pool e, quindi, di essere scelte per generare soluzioni figlie, implementando in tal modo il meccanismo di exploitation.
Una volta che tutti i figli sono stati generati, interviene, secondo una certa probabilità, il processo di mutazione di alcuni dei loro geni (meccanismo di exploration). Terminata la mutazione si ottiene un nuovo insieme di individui (*offspring*) che viene valutato attraverso la funzione di fitness.
Ultimo passo dell'evoluzione è la selezione dei migliori individui tra la popolazione corrente e l'offspring; si parla in genere di *rimpiazzamento* (o *tournament*): solo gli individui più adatti costituiranno la nuova generazione e diventeranno la nuova popolazione corrente. Un'alternativa è rappresentata dalla selezione di un numero fissato di individui migliori della popolazione corrente, mentre gli altri saranno rimpiazzati dai migliori individui dell'offspring. Il processo evolutivo non deve continuare tuttavia all'infinito, ma deve essere generalmente fornito almeno un criterio di terminazione:

- Il più banale è legato ai tempi di calcolo: si può stabilire un numero massimo di generazioni raggiunto il quale si faccia terminare il processo evolutivo; in questo caso l'ottimo fornito dal GA è la soluzione (o le soluzioni) con il fitness più alto all'interno della popolazione corrente;

- Un altro semplice criterio di terminazione può essere posto sul valore di fitness, sia nel caso in cui si conosca il valore in corrispondenza della soluzione ottima globale o si sia in grado di stabilire il valore di una soluzione "accettabile". Utilizzare solo questo criterio di terminazione protrebbe rendere particolarmente lungo il processo evolutivo, oltre a non esistere alcuna garanzia che una delle soluzioni della popolazione raggiunga effettivamente il livello di fitness massimo fissato; in genere questo criterio viene utilzzato in congiunzione ad altri;

- E' possibile usare un criterio di terminazione che fissi un numero massimo di generazioni senza alcun miglioramento nel fitness massimo all'interno della popolazione;

- Ulteriori criteri di terminazione riguardano una percentuale massima di soluzioni con lo stesso livello di fitness o, nel caso degenere, l'omogeneità della popolazione (l'intera popolazione è costituita dalla ripetizione di un unico individuo).

Il criterio di teminazione implementato in un GA può essere uno tra quelli esposti o l'unione di alcuni di essi. Qualunque sia il criterio di terminazione, il processo evolutivo implementato dal GA assicura che la soluzione migliore, su tutto il processo, rimanga all'interno della popolazione all'ultima generazione.

L'operazione di selezione per l'accoppiamento, come è stato già definito, viene effettuata con probabilità legata al fitness. La strategia più conosciuta ed utilizzata è la selezione proporzionale (*Proportional Selection)*, anche nota come "metodo della roulette".
Sia $f(x)$ la funzione di fitness calcolata per un individuo x, si supponga che la popolazione contenga n individui $x_1, \ldots, x_n$, e si indichi con F il *fitness totale della popolazione*:

$$F = \sum_{i=1}^{n} f(x_i)$$

allora, la probabilità di selezionare il cromosoma i è data da:

$$p_i = \frac{f(x_i)}{F}$$

In pratica è come dividere una circonferenza unitaria (roulette) in n settori di diversa ampiezza (Figura 3.2): la dimensione del settore i è pari alla probabilità p_i di selezione del cromosoma x_i. Si estrae, per n volte, un numero casuale nell'intervallo [0,1] ed il cromosoma i corrispondente è inserito nel mating pool; naturalmente i cromosomi con fitness più alto occuperanno settori più ampi nella roulette, manifestando una conseguente maggiore probabilità di essere selezionati per la costituzione del mating pool.
La ricombinazione di una coppia di genitori può essere implementata in modi differenti: in genere si distingue tra *k-point cross-over* (nelle più note versioni *one-point* e *two-point*) e *cross-over uniforme*. Nel k-point cross-over vengono individuate casualmente (e per ogni coppia di genitori selezionati dal mating pool) k posizioni lungo la stringa di bit, dividendo i genitori in $k+1$ porzioni. La coppia di figli è generata prendendo alternativamente le porzioni tra i due genitori; in figura 3.3 sono illustrate le varianti one-point e two-points cross-over. Nel cross-over uniforme, invece, ogni gene del primo figlio ha una diversa probabilità di discendere da uno dei due genitori; il secondo figlio erediterà il gene del genitore non trasmesso al primo discendente. Un'alternativa è rendere indipendente l'ereditarietà tra i figli: ogni gene di ogni figlio discende, con una certa probabilità, da uno dei due genitori. In figura 3.4 è schematizzato il cross-over uniforme, i cui livelli di probabilità sono in genere dell'ordine di 10^{-1}.

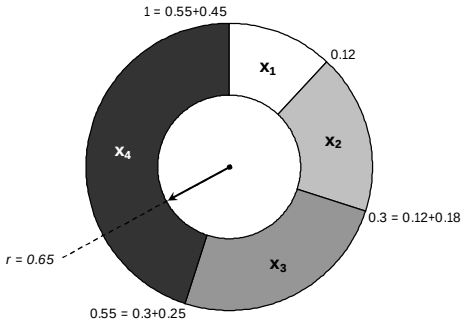

FIGURA 3.2: Roulette

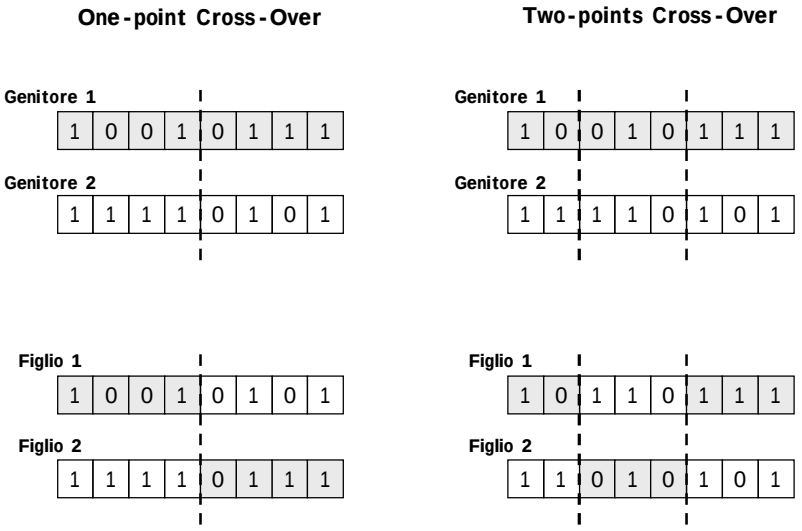

FIGURA 3.3: One-point e Two-point cross-over

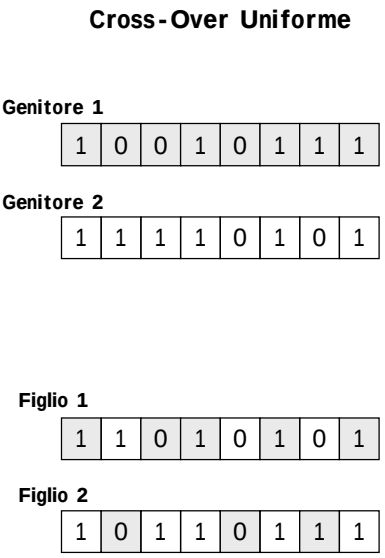

FIGURA 3.4: Cross-Over Uniforme

L'operatore di mutazione agisce in modo molto semplice: per ogni gene di ogni cromosoma figlio viene generato un numero casuale p con distribuzione uniforme nell'intervallo [0;1]. Se $p < p_{fixed}$ il gene viene selezionato per la mutazione. A questo punto bisogna distinguere tra due possibili strategie di mutazione: forte e debole. Nel primo caso il gene selezionato viene costretto alla mutazione ($1 \rightarrow 0; 0 \rightarrow 1$), mentre nel secondo viene scelto casualmente il nuovo valore (0 o 1) da attribuire al gene: in questo caso il valore del gene potrebbe in realtà risultare immutato. I valori tipici per la probabilità di mutazione p_{fixed} variano nell'intervallo [0.001; 0.01].

Esiste inoltre la possibilità di utilizzare anche una codifica a valori reali per i geni, rendendo i cromosomi vettori a componenti reali. Gli operatori di cross-over e mutazione vanno dunque ridefiniti alla luce di questa nuova codifica:

- il *cross-over discreto*, analogo al cross-over uniforme definito per la codifica a bit, funziona nel seguente modo: per ogni gene in posizione i del primo discendente si sceglie, con una certa probabilità fissata p, il genitore che tra i due fornirà il valore per quel gene; il secondo figlio erediterà il valore dell'altro genitore per lo stesso gene. Nel caso in cui $p = 0.5$ il ruolo dei genitori è simmetrico, nessuno prevale sull'altro;

- la tecnica di *cross-over convesso* permette di generare il valore di ogni singolo gene dei cromosomi figli come combinazione convessa dei valori dei due genitori. Siano **u** e **v** i discendenti e **x** e **y** i genitori, allora ogni gene dei figli sarà ereditato come:

$$\mathbf{u}_i = \alpha \mathbf{x}_i + (1-\alpha)\mathbf{y}_i$$
$$\mathbf{v}_i = (1-\alpha)\mathbf{x}_i + \alpha \mathbf{y}_i$$
$$con\ \alpha \in [0;1]$$

L'operatore di mutazione, come per il caso a codifica binaria, può intervenire su un singolo gene o su tutti i geni di un cromosoma. Nel primo caso il gene è scelto casualmente, altrimenti ogni gene può essere rimpiazzato secondo una certa probabilità fissata (in genere uguale per tutti i geni). Anche per la codifica reale è possibile utilizzare meccanismi di mutazione debole o forte. L'unica sostanziale differenza tra la mutazione di cromosomi binari e a valori reali è che nel secondo caso i valori da rimpiazzare saranno numeri reali scelti casualmente all'interno dell'intervallo di valori possibili per ogni gene.

3.3 Lo stato dell'arte per i task di classificazione SVM

3.3.1 Model Selection

Negli ultimi anni l'approccio evolutivo ha acquisito un crecente interesse nel contesto della Model Selection per tecniche di classificazione, tra le quali anche la metodologia SVM. Un esempio è il lavoro di Huang e Wang [74] rappresentato dall'applicazione degli algoritmi genetici per realizzare, contemporaneamente, feature selection ed ottimizzazione dei parametri di un classificatore SVM (più precisamente il parametro γ del kernel RBF ed il parametro di regolazione C). La principale motivazione di relazzare contemporaneamente feature selection e ottimizzazione dei parametri è che l'insieme di feature ha un'influenza sulla scelta dei valori più appropriati per il kernel e viceversa.
Al momento della presentazione del loro lavoro (pubblicato nel 2006) erano già stati proposti metodi di feature selection basati su GA (Raymer, Punch, Goodman, Kuhn e Jain, 2000 [75]; Yang e Honavar, 1998 [76]; Salcedo-Sanz, Prado-Cumplido, Pérez-Cruz e Bousoño-Calzón, 2002 [77]), tuttavia nessuno di essi indirizzava l'aspetto di ottimizzazione dei parametri del classificatore SVM. Il lavoro di Frohlich e Chapelle (2003) [78] proponeva un approccio per la feature selection basato sulla stima dell'errore di generalizzazione, approccio che poteva essere utilizzato anche per l'ottimizzazione del parametro di regolazione C.

Nel loro lavoro, Huang e Wang, propongono una definizione di cromosoma in cui è possibile individuare tre distinte porzioni: la prima composta dai bit relativi al parametro di regolazione C, la seconda relativa al parametro γ del kernel RBF e l'ultima relativa alle features selezionate, come è possibile ossrvare in figura 3.5.

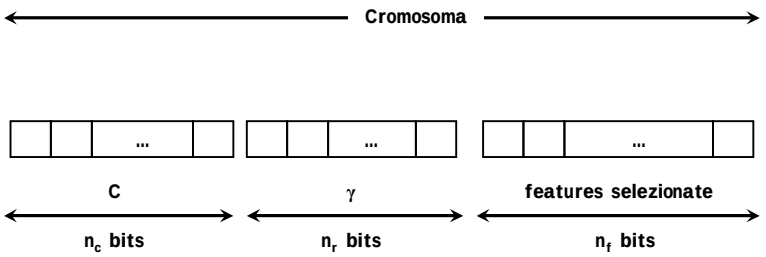

FIGURA 3.5: Definizione di cromosoma per la Model Selection

La rappresentazione classica del genotipo in bit non permette tuttavia una lettura immediata della configurazione del classificatore SVM, è necessario passare al fenotipo per ottenere l'insieme di features selezionate ed i valori dei parametri del classificatore SVM. Gli autori propongono la seguente trasformazione da genotipo a fenotipo, per ogni parametro:

$$p = min_p + \frac{max_p - min_p}{2^l - 1}\, d \tag{3.1}$$

dove p è il fenotipo della stringa in bit, min_p e max_p sono rispettivamente il valore minimo ed il valore massimo per il parametro, ed infine d e l sono rispettivamente il valore decimale e la lunghezza della stringa di bit. I valori minimo e massimo sono definiti dall'utente, la lunghezza in bit della stringa relativa ad ogni parametro (n_c e n_r) influenza direttamente la precisione nella rappresentazione dei valori del parametro stesso. Si può inoltre notare che la 3.1 implementa in realtà una discretizzazione uniforme nell'intervallo $[min_p;\ max_p]$ per ogni parametro, con un numero di intervalli (2^l) dipendente dalla lunghezza in bit della stringa (n_c per il parametro di regolazione C e n_r per il parametro γ del kernel RBF).

Infine, per quanto concerne l'aspetto di feature selection, gli autori utilizzano un bit

(gene) per ogni feature: il valore sarà 1 se la feature è selezionata, 0 altrimenti.
La funzione di fitness definita ed utilizzata dagli autori tiene conto di tre differenti aspetti:

- l'accuratezza
- il numero di features selezionate
- il costo delle features selezionate

in modo che gli individui (configurazioni del classificatore SVM) con maggiore accuratezza, minor numero di features selezionate e minor costo forniscano un più alto valore di fitness. Gli autori utilizzano la seguente definizione:

$$fitness = W_A \, SVM_{accuracy} + W_F \left(\sum_{i=1}^{n_f} C_i F_i \right) \tag{3.2}$$

dove W_A è il peso attribuito all'accuratezza nel calcolo del fitness, $SVM_{accuracy}$ è il valore di accuratezza del classificatore SVM, W_F è il peso relativo al numero di features selezionate, C_i è il costo della feature i ed infine F_i vale 1 se la feature i è selezionata e 0 altrimenti.
I passi dell'approccio proposto da Huang e Wang constitono essenzialmente in:

1. Preprocessare i dati attraverso *scaling*, evitando in tal modo che attributi con un range più ampio dominino su quelli con range più ristretti. Un ulteriore vantaggio dello scaling è di riuscire, in genere, ad incrementare l'accuratezza del classificatore SVM sull'insieme di addestramento;
2. Convertire, per ogni parametro SVM (C e γ) il genotipo in fenotipo, al fine di ottenere la configurazione del classificatore SVM identificato da uno specifico cromosoma della popolazione;
3. Identificare l'insieme di features selezionate, informazione indicata in ogni cromosoma secondo la codifica precedentemente descritta;
4. Valutare il fitness di ogni soluzione all'interno della popolazione corrente. Per ogni cromosoma, una volta convertito il genotipo in fenotipo per ogni parametro e dopo aver identificato l'insieme delle features selezionate, viene addestrato lo specifico classificatore SVM sul relativo insieme di addestramento, mentre un insieme di validazione viene utilizzato per stimare l'accuratezza del modello. Infine, viene calcolato il valore di fitness secondo la 3.2;

5. Verificare se i criteri di terminazione sono soddisfatti: in caso affermativo il processo termina, altrimenti si passa ad una nuova generazione utilizzando lo schema classico di un GA, con le sue operazioni di selezione, cross-over, mutazione e rimpiazzamento.

Gli esperimenti eseguiti da Huang e Wang hanno previsto l'utilizzo di ben 11 dataset dal database UCI (www.ics.uci.edu/˜mlearn/ MLRepository.html). Il calcolo dell'accuratezza è stato eseguito attraverso tecnica di validazione *10fold-cross* e sono stati messi a confronto i valori di fitness ottenuti dalla soluzione proposta dagli autori e dal classico approccio a griglia, variando i valori di C e γ.
Su tutti gli 11 dataset la soluzione proposta ha riportato valori di fitness migliori rispetto all'approccio a griglia, ciò è principlamente dovuto al fatto che la tecnica degli autori realizza anche la selezione delle features, aspetto non considerato nell'approccio a griglia, per il quale l'addestrameno dei classificatori è stato eseguito su tutte le features dei vari insiemi di addestramento.

3.3.2 Multiple Kernel Learning

Un interessante esempio di applicazione degli algoritmi genetici per la costruzione del kernel ottimale di un classificatore SVM (Multiple Kernel Learning) è il lavoro di Lessmann, Stahlbock e Crone [79]. Partendo dall'assunzione che se $K1$ e $K2$ sono kernel validi allora anche $K1 + K2$ e $K1 * K2$ risulteranno kernel validi, gli autori hanno selezionato un insieme di kernel di base da combinare attraverso GA. In particolare hanno selezionato i seguenti 5 kernel:

- RBF (K_{rad}) $K_{rad}(x_i, x_j) = \exp(-\alpha \left\|x_i - x_j\right\|^2)$;
- Polinomiale (K_{poly}) $K_{poly}(x_i, x_j) = (\alpha(x_i \cdot x_j) + \beta)^d$;
- Sigmoidale (K_{sig}) $K_{sig}(x_i, x_j) = \tanh(\alpha(x_i \cdot x_j) + \beta)$;
- Anova (K_{anova}) $K_{anova}(x_i, x_j) = \left(\sum_j \exp(-\alpha(x_i - x_j))^2\right)^d$;
- Multiquadratico inverso (K_{imq}) $K_{imq}(x_i, x_j) = \frac{1}{\sqrt{\left\|x_i - x_j\right\|^2 + \beta^2}}$;

Il kernel combinazione è ottenuto come:

$$\tilde{K} = K_{poly}^{K_1} \otimes_1 K_{rad}^{K_2} \otimes_2 K_{sig}^{K_3} \otimes_3 K_{imq}^{K_4} \otimes_4 K_{anova}^{K_5}$$

con $\otimes_j \in +; \cdot \forall j = 1, \ldots, 4$.
L'approccio proposto cerca di ottimizzare contemporaneamente differenti parametri della

combinazione kernel: gli esponenti dei kernel $(K_1, \ldots, K_5)$, gli operatori di combinazione $(\otimes_1, \ldots, \otimes_4)$ ed infine i parametri (α, β, d) dei singoli kernel sopra definiti.
L'approccio GA proposto dagli autori differisce in modo sostanziale da quello classico che vede ogni singolo cromosoma rappresentato come sequenza di bit (come descritto anche nella precedente sezione 3.3.1, in merito al lavoro di Huang e Wang). Gli autori utilizzano infatti 5 geni a valori interi per rappresentare gli esponenti dei kernel, 4 geni binari (gli unici con rappresentazione a bit) per rappresentare gli operatori di combinzione tra kernel (somma o prodotto), 15 geni a valori reali per rappresentare i parametri dei vari kernel ed infine un singolo gene a valori reali per rappresentare il parametro di regolazione C; il cromosoma definito nel lavoro di Lessmann, Stahlbock e Crone è riportato in figura (3.6)

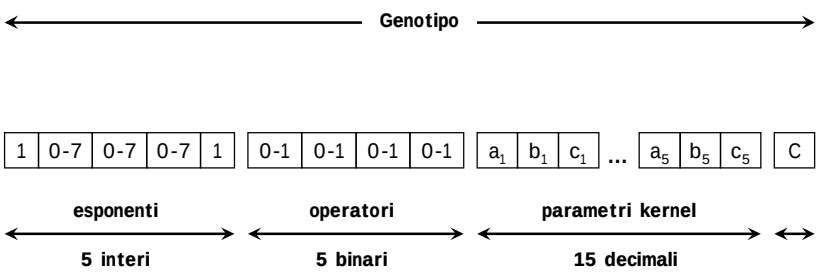

FIGURA 3.6: Definizione di cromosoma per il Kernel Learning

Deve essere inoltre sottolineato che per i kernel *polinomiale* e *anova* il gene relativo all'esponente del kernel è in realtà obsoleto, visto che un esponente compare già nella definizione dei due stessi kernel ed è considerato quindi parametro interno del kernel.
Altra considerazione importante in merito all'approccio proposto dagli autori è la loro idea di considerare la feature selection come un'operazione di pre-processamento dei dati non (inter)dipendente dal processo di ottimizzazione del kernel, come al contrario era stato considerato da Huang e Wang nella loro soluzione proposta per la Model Selection (sezione 3.3.1). In realtà, a seconda che la feature selection venga eseguita indipendentemente o meno dall'algoritmo di apprendimento, le tecniche di feature selection possono essere distinte in approcci *wrapper* e *filter*: la soluzione proposta da Huang e Wang per

la Model Selection prevede l'uso dell'approccio *wrapper*, mentre quella proposta da Lessmann, Stahlbock e Crone per il Kernel Learning prevede l'uso dell'approccio *filter*.
Un ultimo aspetto è relativo alla costruzione della funzione di fitness, punto cruciale su cui si basa il meccanismo di evoluzione delle soluzioni. Gli autori hanno definito due diverse funzioni di fitness, la prima corrispondente al valore di "accuratezza bilanciata" (*Balanced Classification Accuracy - BCA*) e la seconda basata su un upper-bound dell'errore leave-one-out:

$$T = \frac{n_{SV}}{\ell}$$

con n_{SV} numero di support vectors e ℓ numero di istanze di addestramento.
L'accuratezza bilanciata (BCA) è in pratica la media delle accuratezze rilevate su ogni classe:

$$BCA = \frac{\sum_{i=1}^{c} \frac{p_i}{n_i}}{c} \tag{3.3}$$

con c il numero di classi, p_i il numero di classificazioni corrette per la classe i e n_i il numero di istanze della classe i nell'insieme di addestramento.
Gli autori confrontano i valori di fitness forniti dal loro approccio con quelli ottenuti da una sperimentazione a griglia su 4 dataset provenienti dal progetto Statlog e dal database UCI (www.ics.uci.edu/~mlearn/ MLRepository.html). In tutti i casi la soluzione proposta ha fornito prestazioni migliori rispetto al classico approccio a griglia, per entrambe le definizioni della funzione di fitness.

3.3.3 SVM Ensemble

Un interessante e recente lavoro di ricerca, di Ling-Min He et al [80], studia i possibili vantaggi offerti dall'applicazione degli algoritmi genetici per la realizzazione di un sistema di combinazione di classificatori SVM.
Gli autori prendono in considerazione differenti metodi di combinazione per SVM proposti negli anni: bagging, boosting, modelli decisionali a SVM multiple (MSDM - *Multiple SVM Decision Model*) e a SVM eterogenee (HSDM - *Heteogeneous SVM Decision Model*), confrontando tali metodologie su 4 datasets multiclasse. Inoltre propongono, per bagging e boosting, l'utilizzo degli algoritmi genetici per ottimizzare i pesi di combinazione dei classificatori SVM basali.
Bagging e boosting sono metodologie di combinazione generali già trattate nella sezione 1.2 del presente lavoro di tesi. MSDM è invece una tecnica proposta in [81, 82] per migliorare le prestazioni e l'affidabilità dei classificatori SVM. La strategia consiste nel dividere l'intero insieme di addestramento in più sottoinsiemi della stessa dimensione ed addestrare, su ogni sottoinsieme, un classificatore SVM. Tutti i modelli decisionali ottenuti vengono infine combinati tra loro attraverso un semplice schema di voto di

maggioranza. La tecnica HSDM, infine, propone di addestrare differenti SVMs multi-classe sull'insieme di addestramento, che è lo stesso per tutti i classificatori a differenza delle altre metodologie. Un'opportuna strategia di aggregazione viene poi utilizzata per combinare tra loro i modelli decisionali dell'insieme. A causa delle differenti strategie implementative, le conclusioni sono piuttosto controverse: boosting-SVM ha riportato le prestazioni migliori in [83], ma anche le peggiori in [84], mentre MSDM ha fornito buoni risultati in [82].
Gli autori propongono il seguente schema procedurale, inerente l'applicazione dei GA al boosting ed al bagging per l'ottimizzazione dei pesi di uno schema di combinazione a maggioranza-pesato:

Input: insieme di addestramento Z, insieme di validazione Z_V, algoritmo di apprendimento SVM L, numero di tentativi K, soglia λ

- Addestramento -
for k = 1 to K
{
Z_k = insieme ottenuto da Z con bagging o boosting;
$f_k = L(Z_k)$;
}
Generare una popolazione di vettori di pesi.
Evolvere la popolazione sulla base della funzione di fitness $f(\mathbf{w}) = E^V_{\mathbf{w}}$, dove $E^V_{\mathbf{w}}$ è la stima dell'errore di generalizzazione dell'individuo (soluzione) $\mathbf{w}$ sull'insieme di validazione Z_V.
$\mathbf{w}^*$ è il miglior vettore di pesi al termine dell'evoluzione.

- Test della combinazione f^* -

$$f^*(x) = \arg\max_{y \in Y} \sum_{\mathbf{w}^*_k > \lambda : f_k(x) = y} \mathbf{w}_k f_k(x)$$

I risultati delle sperimentazioni eseguite dagli autori hanno evidenziato che, per quanto concerne le tecniche di combinazione note, HSDM fornisce prestazioni leggeremente migliori dei singoli classificatori basali su 3 dei 4 datasets, mentre boosting raggiunge i livelli di accuratezza più alti su tutti i datasets. Infine, le strategie boosting+GA e bagging+GA proposte dagli autori forniscono un ulteriore miglioramento rispetto allo stesso boosting, ed inoltre la versione boosting+GA risulta migliore della bagging+GA.

Capitolo 4

La Soluzione Proposta

4.1 Variazioni alla formulazione classica di un Algoritmo Genetico

L'implementazione del Framework di classificazione ad alto livello ha richiesto l'apporto di alcune modifiche alla formulazione canonica dell'approccio GA, alla luce delle problematiche e delle considerazioni esposte nei capitoli precedenti.
Uno dei primi punti che si è ritenuto cruciale affrontare è l'eliminazione di un problema che in questo lavoro è indicato con il nome *mutazione implicita*. La probabilità di mutazione è un parametro in genere fissato dall'utente, la scelta fatta non è tuttavia esattamente rispettata nel momento in cui l'operazione di cross-over, su cromosomi con rappresentazione a bit, divida la stringa binaria relativa ad una particolare componente della soluzione; in tal caso la trasmissione del valore della componente ai figli può presentare valori già mutati rispetto a quelli dei genitori. In figura 4.1 è schematizzato il fenomeno di mutazione implicita, fenomeno completamente casuale.
Un'ulteriore modifica riguarda la trasformazione da genotipo a fenotipo: formule come quella prevista nel lavoro di Huang e Wang (equazione 3.1) utilizzano meccanismi di discretizzazione uniforme che generano un insieme di valori "equidistanti" nel range di variabilità di ogni parametro. Questo può essere utile nel momento in cui non si sia interessati a testare specifiche configurazioni, ma solo a ricercare una soluzione sufficientemente buona, le cui componenti abbiano valori compresi all'interno di predeterminati range.
In analogia alla sperimentazione a griglia, tuttavia, è frequente la situazione in cui si vuole mantenere una sorta di controllo, di scelta, sulle configurazioni da testare, selezionando un insieme di valori possibili per ogni componente della soluzione. A tal proposito, nella rappresentazione a bit interviene l'ulteriore problematica di dimensionamento delle

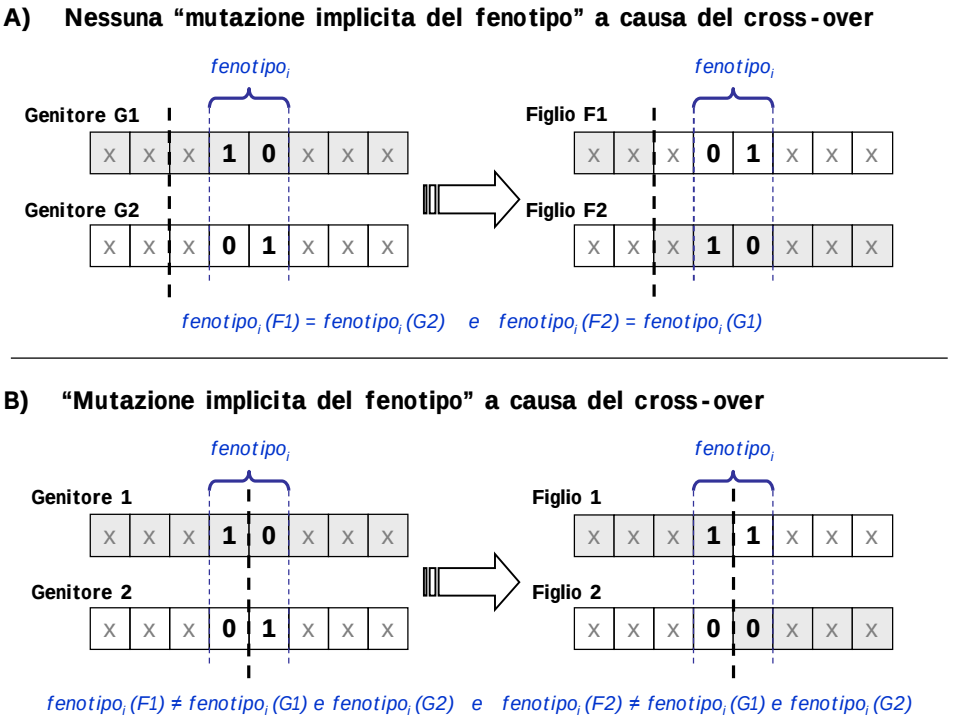

FIGURA 4.1: Fenomeno di mutazione implicita del fenotipo

stringhe di bit. Si supponga infatti di aver selezionato 9 valori di interesse per una componente della soluzione: sapendo che con n bit si possono rappresentare al massimo 2^n valori, per codificare i 9 valori sono necessari almeno 4 bits ($2^3 = 8$ e $2^4 = 16$) che sono tuttavia più bits del necessario; ben 7 possibili stringhe di bit del genotipo non hanno infatti un corrispettivo nel fenotipo. Non è consigliabile attribuire le rimanenti stringhe solo ad alcuni valori del fenotipo poiché ciò inciderebbe direttamente sulla probabilità di presentarsi delle varie configurazioni.
Tale problematica è ancora più apprezzabile quando per alcune componenti della soluzione non è possibile aumentare il numero di valori del fenotipo fino ad un numero rappresentabile esattamente con n bits: per un parametro numerico è semplice ampliare il range o inserire direttamente altri valori da testare, anche se non di particolare interesse, mentre tale procedimento non può essere attuato nel momento in cui la componente sia di natura puramente nominale; si pensi ad esempio al tipo di kernel: si supponga di voler utilizzare tutti i kernel basali conosciuti in un preciso momento storico, e che essi siano in numero m tale che non esista un intero l per il quale si abbia $2^l = m$, allora non è naturale stabilire un criterio sulla base del quale scegliere i kernel da non considerare nel processo di ricerca della soluzione ottima.

La rappresentazione dei cromosomi proposta nel presente lavoro abbandona la classica espressione in bit e lavora direttamente con il fenotipo delle soluzioni: la modalità di

rappresentazione è molto simile a quella presentata da Lessmann *et al* nel loro lavoro sul Multiple Kernel Learning (sezione 3.3.2), indirizzando inoltre le specifiche problematiche fin qui evidenziate. Un'altra importante caratteristica della rappresentazione proposta è di abbandonare il classico concetto di "gene": lavorando direttamente con il fenotipo delle soluzioni, ovvero con i valori delle componenti, si ritiene preferibile utilizzare il concetto di "carattere", soffermandosi di conseguenza sulla trasmissione da genitori a figli di quei caratteri (componenti delle soluzioni) che caratterizzano gli individui più adatti. Per lavorare direttamente con il fenotipo delle soluzioni è inoltre necessario gestire ogni componente nel modo più opportuno secondo la propria natura: nel seguito si fornirà la definizione di carattere "discreto" e "continuo".

4.1.1 La codifica a "caratteri" genetici

Il problema della mutazione implicita, nel quale si incorre utilizzando la classica rappresentazione in bit per i cromosomi, può essere facilmente risolto imponendo che l'operazione di cross-over non "tagli" il corredo genetico in punti tali da alterare i valori del fenotipo nei figli. La problematica della mancata corrispondenza tra il numero di configurazioni in bit di un parametro ed i valori che per esso vogliono essere testati non è tuttavia indirizzato da tale semplice modifica nell'operazione di cross-over.
La soluzione proposta prevede di lavorare con una codifica mista, a "caratteri discreti" e "continui", i cui valori coincidono con quelli assunti dalle componenti della soluzione. In termini semplicistici, si lavora direttamente con il fenotipo senza passare per la codifica a bit (genotipo).
Si prenda il caso di una semplice SVM di cui si voglia trovare una configurazione ottimale (secondo una certa funzione di fitness) andando ad ottimizzare:

- il parametro di regolazione C, che varia in modo continuo nell'intervallo [1;10],
- il tipo di kernel: RBF, polinomiale normalizzato o nessun kernel (SVM lineare),
- il parametro interno del kernel: γ nel caso di kernel RBF (che varia in modo continuo nell'intervallo [0.001; 1000]) o il grado (*degree*) per il kernel polinomiale (che può assumere 5 valori interi);

Questo problema di Model Selection prevede l'uso di componenti di differente natura: continue (parametro di regolazione C e γ per il kernel RBF), strettamente nominali (tipo di kernel) ed interi (grado del kernel polinomiale). Gli individui della popolazione avranno quindi determinati "caratteri" che vengono trasferiti da genitori a figli con modalità differenti a seconda della loro stessa natura.

I caratteri "discreti" vengono trasmessi in modo inalterato ai figli: ognuno dei 2 discendenti eredita, in modo indipendente dall'altro, il carattere di uno dei due genitori (ad esempio il caso di trasmissione del tipo di kernel); i caratteri "continui" vengono trasmessi attraverso una "miscelazione", ad esempio come combinazione convessa tra i caratteri dei due genitori (nel caso della trasmissione del parametro di regolazione C e del parametro γ del kernel RBF).

Ragionando per analogia, è lo stesso processo della reale trasmissione dei caratteri a seconda del loro tipo di *variabilità*: discontinua o continua. La prima è quella che prevede un numero finito di valori (fenotipi) per il carattere, un esempio è il colore di occhi di *Drosophila melanogaster* (più comunemente noto come moscerino della frutta o dell'aceto) i cui possibili fenotipi sono il rosso, il bianco e l'arancio-albicocca; il secondo tipo di variabilità riguarda invece tutti quei caratteri con un elevato (e idealmente infinito) numero di fenotipi, come l'altezza o il colore della pelle nell'essere umano.

Nel lavoro si ritiene che la rappresentazione proposta sia più compatta e comprensibile, flessibile, e soprattutto, eliminando il fenomeno di mutazione implicita del fenotipo, scinda completamente la mutazione dal cross-over (come teoricamente dovrebbe essere), in modo da offrire all'utente la possibilità di controllare più accuratamente il processo evolutivo attraverso una corretta impostazione della probabilità di mutazione.

In figura 4.2 sono riportati due esempi di definizione di cromosoma per Model Selection: nella prima le soluzioni sono definite da soli caratteri discreti (in stretta analogia con la classica sperimentazione a griglia), nella seconda compaiono invece contemporaneamente caratteri discreti e continui; ciò è dovuto all'intrinseca natura categoriale di alcuni caratteri, ovvero tipo di kernel e grado del kernel polinomiale (in grigio in figura).
La seconda definizione permette di far "esplodere" lo spazio di ricerca delle soluzioni, permettendo di selezionare qualsiasi valore reale all'interno del range specificato per ognuno dei caratteri continui sopra menzionati. In realtà tale potenzialità è comunque limitata dal calcolatore, che naturalmente non può scendere oltre una certa precisione. In merito a tale considerazione, nell'implementare il Framework si è deciso di lasciare all'utente la possibilità di scegliere il livello di precisione per ogni carattere continuo; ciò equivale a discretizzare il range di ogni parametro, tornando ad una situazione di tipo classico, per esempio: si supponga di stabilire che il parametro di regolazione C vari nell'intervallo [1;10] e si fissi inoltre il livello di precisione al centesimo (due cifre decimali). Il numero di valori che effettivamente il parametro potrà assumere sono $(10 - 1)/0.01 = 900$; è importante notare che gestire 900 valori con la rappresentazione a bit dei cromosomi richiederebbe una stringa di lunghezza $\lceil \log_2 900 \rceil$, ovvero 10 geni per il solo parametro C, con inoltre un eccessivo numero configurazioni di bit non utilizzabili ($2^{10} = 1024 > 900$).

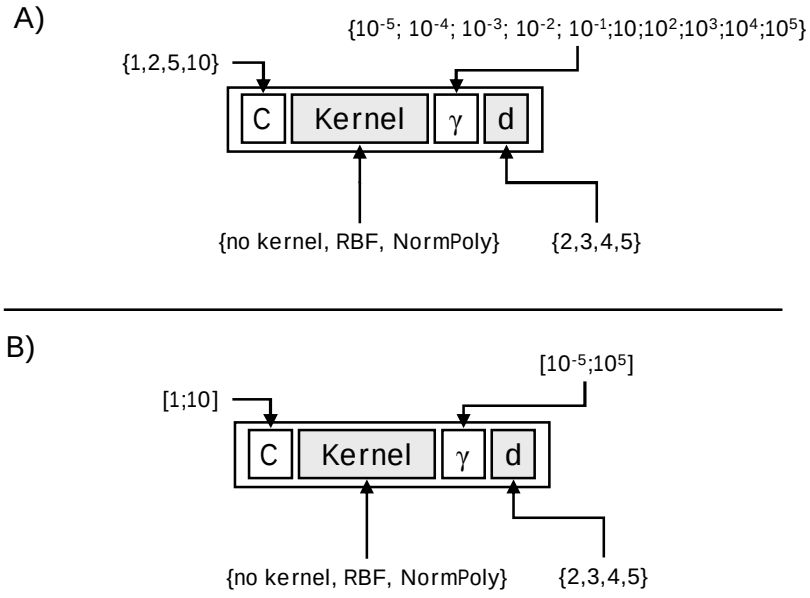

FIGURA 4.2: Esempi di cromosoma per Model Selection a caratteri A) discreti e B) continui

4.1.2 Cross-over e Mutazione

Come già accennato, la differente natura dei caratteri delle soluzioni prevede che venga attuata la più opportuna gestione per ognuno di essi, ciò si traduce in una ridefinizione degli operatori genetici di cross-over e mutazione.

Per caratteri discreti il cross-over può intervenire secondo due differenti modalità, a seconda che il processo di ereditarietà sia indipendente o meno tra i figli: il primo discendente eredita il carattere di uno dei due genitori, il secondo eredita quello dell'altro genitore (ereditarietà dipendente tra figli) o quello di uno dei due genitori (ereditarietà indipendente tra figli).

Per quanto riguarda i caratteri continui invece, il cross-over è semplicemente il noto cross-over convesso definito per i cromosomi a geni continui. In realtà anche l'operatore definito per i caratteri discreti potrebbe essere assimilato al cross-over uniforme per cromosomi a geni continui, tuttavia esiste una significativa differenza rispetto alla rappresentazione classica: quest'ultima prevede infatti che l'operatore di cross-over sia unico (uniforme o convesso) e non differenziato per ogni gene. Altre differenze sono apprezzabili nel momento in cui si considera l'implementazione del processo evolutivo dei caratteri nel suo complesso, ovvero come combinazione degli effetti di entrambi gli operatori genetici, cross-over e mutazione.

La mutazione per i caratteri discreti è infatti definita in modo tale che il gene selezionato per l'operazione assuma uno dei possibili valori scelti a caso tra quelli specificati, a differenza dei caratteri continui che assumono invece un qualsiasi valore reale all'interno di un intervallo prefissato. Gli aspetti generali della mutazione sono stati mantenuti: l'operatore può, a scelta, interessare uno solo dei geni di ogni cromosoma o più di uno, e può inoltre essere di tipo debole o forte.
La combinazione degli operatori di cross-over e mutazione permette di far comprendere meglio le ragioni delle due differenti definizioni di carattere: un individuo può trasmettere i suoi caratteri discreti ad un suo figlio, o eventualmente ad entrambi, in modo inalterato, successivamente questi potranno subire, secondo una certa probabilità fissata, una mutazione, modificandosi in altre possibili manifestazioni prestabilite per quei caratteri. Lo stesso individuo può inoltre contribuire, in combinazione con l'altro genitore, alla trasmissione dei caratteri continui ai figli, successivamente questi potranno essere mutati in valori casuali compresi negli intervalli reali definiti per quei caratteri.

4.2 Le popolazioni del Framework di classificazione ad alto livello

L'implementazione del Framework prevede l'uso di 3 differenti popolazioni relative alle 3 differenti tematiche indirizzate: la *Model Selection*, il *Multiple Kernel Learning* ed il *Sistema di Combinazione di classificatori SVM*. In figura 4.3 sono schematizzate le definizioni dei cromosomi delle 3 differenti popolazioni.
Il cromosoma per la Model Selection è stato già illustrato in precedenza, qui si vuole solo precisare che il numero di configurazioni possibili risulta pari a $|C| \times (|\gamma| + |degree| + 1)$, con $|.|$ il numero di valori possibili nel caso di caratteri discreti o il numero di valori reali plausibili nel caso di caratteri continui, dati range di variabilità e precisione.
Per quanto concerne la definizione di cromosoma per il Sistema di Combinazione di N classificatori SVM si può osservare come siano presenti un carattere per ognuno degli N classificatori: N caratteri per i pesi di combinazione ($\mathbf{w}$), N per i parametri di regolazione C, N per i tipi di kernel, N per gli eventuali parametri γ o d (rispettivamente per i kernel RBF o polinomiali). In questo caso il numero di configurazioni risulta pari a $(|\mathbf{w}| \times |C| \times (|\gamma| + |degree| + 1))^N$, esponenziale rispetto al numero N di SVMs che si desidera combinare.
Infine, in merito alla definizione di cromosoma per il Kernel Learning, come combinazione di M kernel basali, si nota: un solo carattere per il parametro di regolazione C, M caratteri per i coefficienti di combinazione a, M per i tipi di kernel, M per gli eventuali parametri γ o d (rispettivamente per kernel RBF o polinomiali). A differenza del lavoro

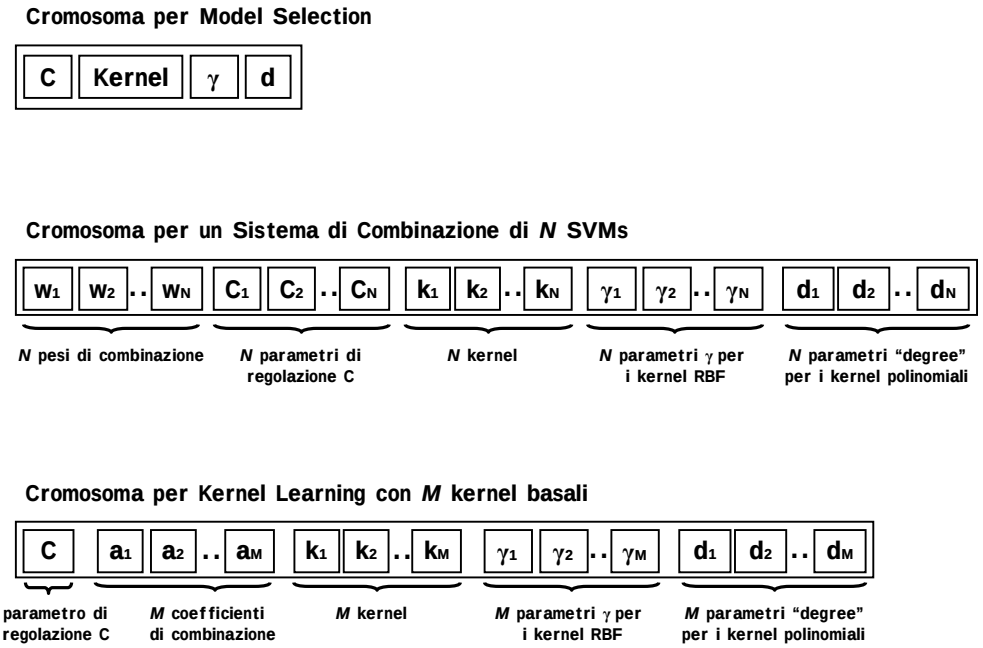

FIGURA 4.3: Definizione dei cromosomi delle 3 popolazioni per il Framework proposto

di Lessmann (3.3.2), in questa definizione non sono considerati gli operatori di combinazione (somma o prodotto) ma viene fissato a priori l'operatore di somma (combinazione lineare di kernel); la definizione del cromosoma può essere facilmente modificata senza alcun onere dal punto di vista implementativo, aumenterebbe tuttavia il numero di configurazioni possibili. Secondo la definizione proposta, il numero di configurazioni risulta pari a $(|C| \times (|a| \times (|\gamma| + |degree| + 1)))^M$, che risulta minore di quello per la combinazione di SVMs se si fissa $N = M$. E' fondamentale sottolineare un'importante differenza tra la definizione di cromosoma qui proposta per implementare il Multiple Kernel Learning rispetto a quella adottata da Lessman *et al* nel loro lavoro (sezione 3.3.2): nella prima anche il tpo di kernel è una variabile da ottimizzare mentre Lessmann prevede una combinazione fissa di 5 tipi di kernel. L'approccio qui proposto, al contrario, fornisce un insieme di kernel basali ma non pone alcun vincolo sulla scelta per la combinazione, in modo tale che il kernel risultante possa essere anche una combinazione di M kernel dello stesso tipo con valori differenti dei parametri.

4.3 Dettagli Implementativi

La soluzione proposta è stata interamente implementata in Java attraverso l'uso dell'ambiente di sviluppo (IDE) *Eclipse* (versione 3.4.1). L'implementazione ha riguardato non solo la realizzazione del framework ad alto livello basato su GA, ma anche su Tabu Search e Ant Colony Optimization (trattate nel capitolo 5). Nonostante il proliferare di strumenti software open-source, specialmente in relazione alla metodologia GA, si è preferito sviluppare direttamente una libreria di classi Java, avendo così il pieno controllo su progettazione ed implementazione delle variazioni proposte alla formulazione GA classica, senza andare ad intaccare e modificare (molto probabilmente con maggiore difficoltà) codici già progettati da altri soggetti. Tale necessità è risultata ancora più forte in merito al Multiple Kernel Learning basato su strategia ACO.
I software implementati sono da ritenere "grezzi" e sviluppati strettamente per scopi di ricerca, di conseguenza necessitano senza dubbio di un'ulteriore fase di ingegnerizzazione per essere presentati come strumenti di ricerca stabili ed affidabili. Una fase di progettazione più accurata ha già avuto inizio ed è stata rivolta esclusivamente alla realizzazione di una libreria di classi Java per l'approccio GA proposto nel lavoro di tesi, anche in visione dei promettenti risultati ottenuti e presentati nel capitolo 6. L'implementazione della meta-euristica di ricerca GA è stata realizzata in modo completamente indipendente dallo specifico algoritmo di addestramento SVM che si voglia usare, l'unico passo richiesto è l'implementazione di un sistema di una o più classi volte a realizzare i meccanismi di interfaccia con l'utente e di integrazione tra meta-euristica di ricerca e addestramento-validazione dei classificatori SVM.
In figura 4.4-A è schematizzata l'architettura generale del sistema software per il framework di classificazione ad alto livello basato sulla libreria Java realizzata. In grigio, nella figura, sono indicati: un generico strumento di apprendimento SVM e il modulo software da realizzare sia per l'interfacciamento con l'utente che per l'integrazione tra la meta-euristica di ricerca e lo strumento di apprendimento SVM.
Un'istanza di tale sistema è stata realmente sviluppata, usando WEKA (*Waikato Environment for Knowledge Analysis*) come strumento per l'apprendimento SVM e realizzando inoltre un'interfaccia grafica per l'utente del sistema (*GAFrameworkGUI 1.0*); in figura 4.4-B è illustrato il sistema realizzato (completamente Java-based).

In figura 4.5 è riportata la schermata iniziale dell'interfaccia utente. Attraverso il bottone "load training set" è stato caricato un insieme di addestramento (file nel formato ARFF previsto da WEKA) e, nell'area di testo, sono riportate alcune informazioni riassuntive sulle principali caratteristiche del dataset, aggiornate dinamicamente in base a quale attributo, tra quelli nominali, venga selezionato come attributo di classe. E' inoltre possibile impostare il tipo di validazione da eseguire e la funzione per la valutazione del

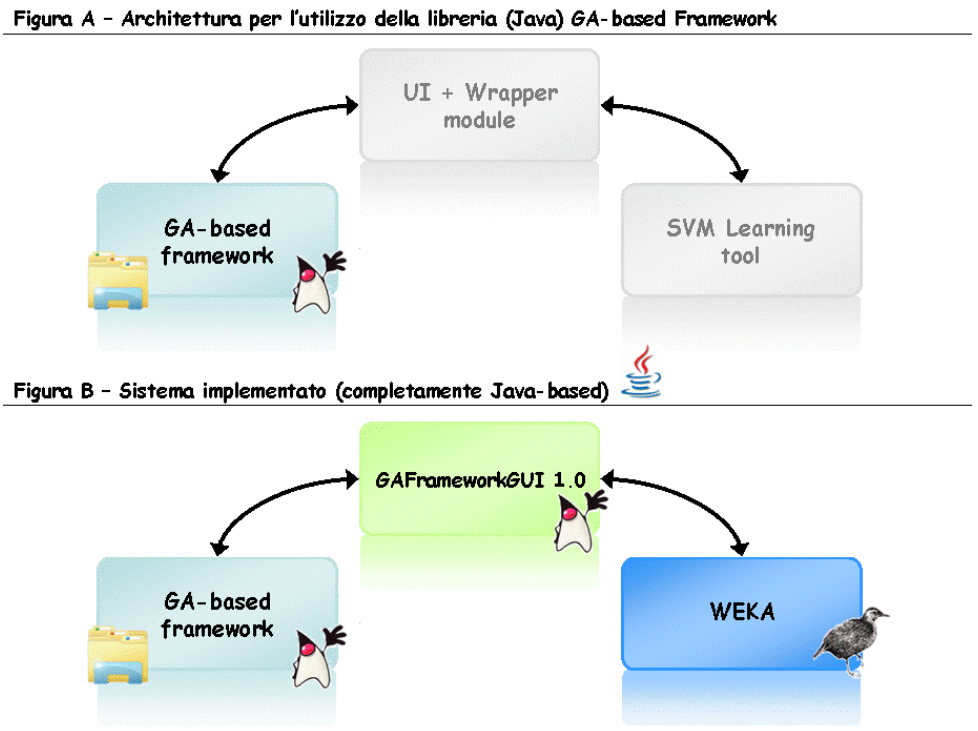

FIGURA 4.4: Framework di classificazione ad alto livello: architetture di un generico sistema (A) e di una reale implementazione realizzata (B)

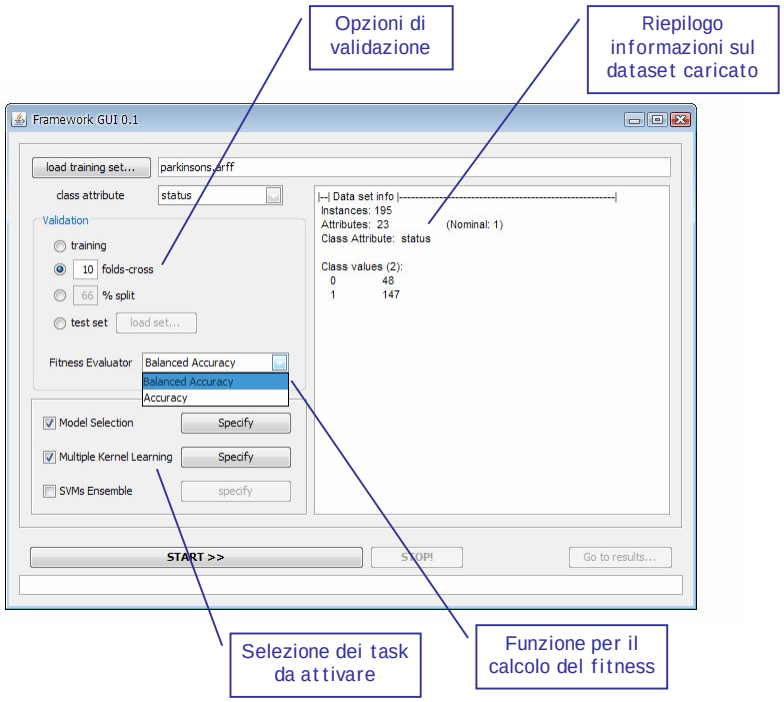

FIGURA 4.5: GAFrameworkGUI 1.0 - Schermata principale

fitness; attualmente sono state implementate le due più note ed utilizzate misure di affidabilità: l'accuratezza e l'accuratezza bilanciata. Infine, è possibile scegliere di eseguire tutti i task del framework o soltanto uno o due di essi (Model Selection, Multiple Kernel Learning e combinazione di più SVMs).
Una volta selezionato un task è necessario specificare la configurazione della relativa popolazione. In figura 4.6 è riportata la schermata che permette di configurare la popolazione per il task di Model Selection: è possibile impostare la dimensione della popolazione, i criteri di terminazione e tutti i parametri relativi al classificatore SVM. In

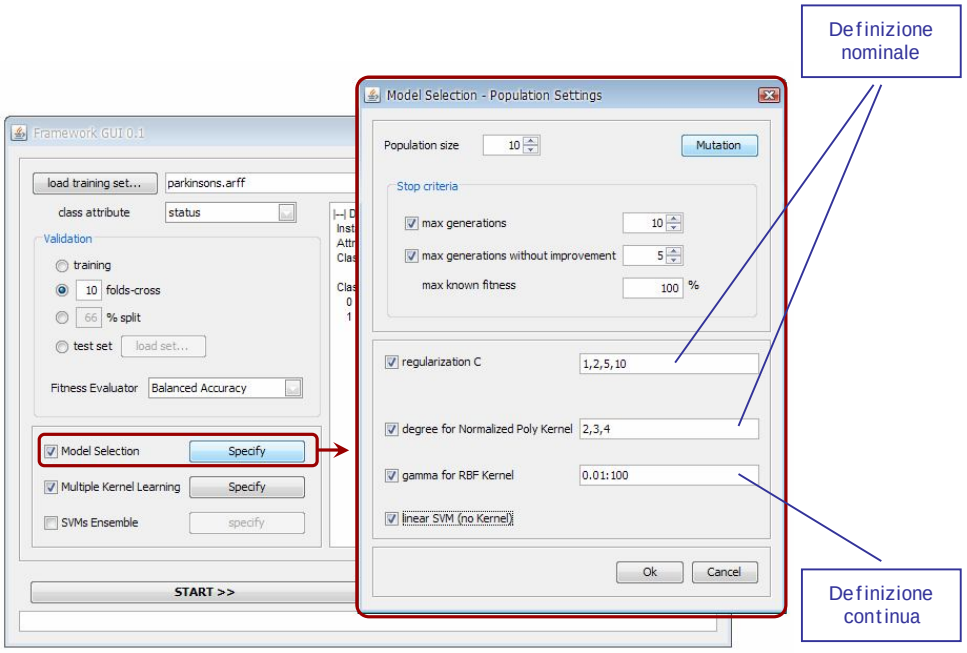

FIGURA 4.6: GAFrameworkGUI 1.0 - Popolazione per Model Selection

figura è inoltre possibile notare i due diversi tipi di definizione delle componenti delle soluzioni: valori separati da virgole per il caso di componente nominale (in figura, il grado del kernel polinomiale e il parametro di regolarizzazione C) e 2 valori separati dal simbolo ":" che rappresentano gli estremi, rispettivamente minimo e massimo, del range di variabilità per le componenti continue. Va sottolineato che la configurazione è altamente personalizzabile, potendo selezionare anche solo un sottoinsieme dei parametri del classificatore SVM elencati. Infine, bisogna tener conto che, nel caso in cui più kernel vengano selezionati, un'ulteriore componente delle soluzioni è proprio il tipo di kernel.

Nella seguente figura 4.7 è invece mostrata la schermata per configurare il task di Multiple Kernel Learning. Valgono sostanzialmente le stesse indicazioni fornite precedentemente

in merito alla Model Selection, la differenza sostanziale risiede nella definizione dei parametri kernel e dei coefficienti della combinazione: è infatti necessario specificare tante righe quanti sono i kernel della combinazione (cfr. sezione 4.2). Una modalità alternativa consisterebbe nel far impostare all'utente il numero di kernel da combinare ed utilizzare un'unica definizione per ogni tipologia di kernel, ma si è deciso di utilizzare la modalità di definizione "una riga per ogni kernel" in modo tale da lasciare all'utente la libertà di utilizzare anche parametri differenti per kernel dello stesso tipo. Si supponga ad esempio di impostare 2,3,4 come gradi dei primi due kernel polinomiali e 2,3,4,5,6,7 per il terzo, questo tipo di definizione stabilisce che se nella soluzione i primi due kernel della combinazione sono polinomiali il loro grado sarà un valore intero tra 2 e 4, mentre per il terzo sarà un valore intero tra 2 e 7.

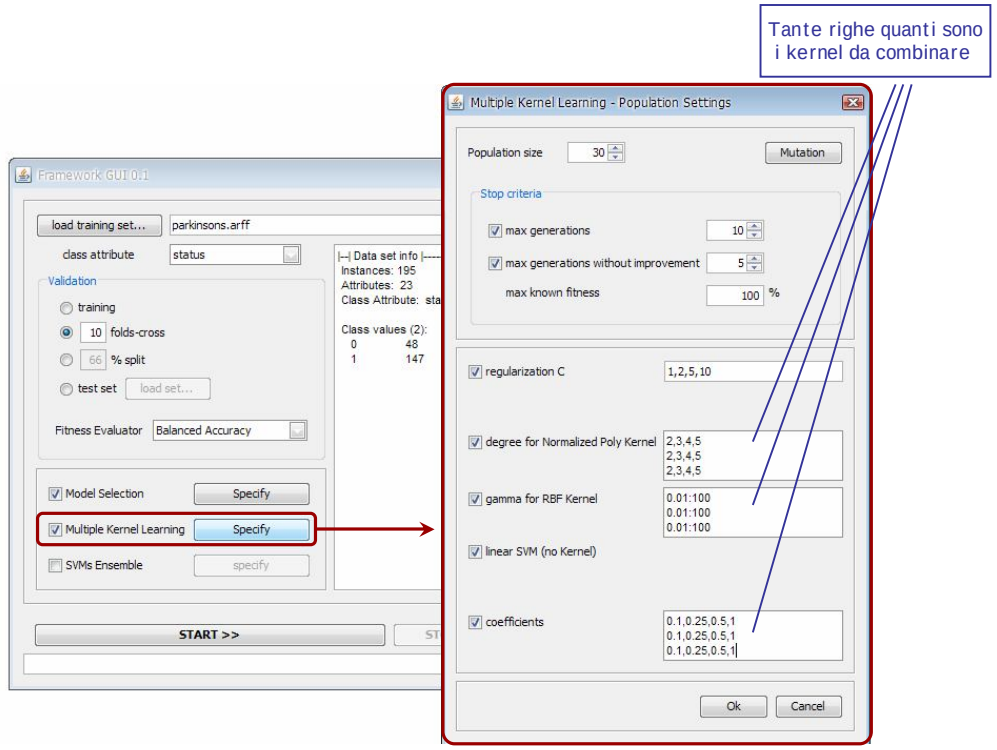

FIGURA 4.7: GAFrameworkGUI 1.0 - Popolazione per Multiple Kernel Learning

Nella successiva figura 4.8 è riportata la schermata per la configurazione del sistema di combinazione di SVMs. In questo caso la schermata è molto simile a quella per il Multiple Kernel Learning, l'unica differenza riguarda la definizione del parametro di regolarizzazione C, in questo caso tante righe quante sono le SVMs che si desidera combinare (mentre per il Multiple Kernel Learning il paramaetro di regolarizzazione C è unico in quanto è unica la SVM, e la combinazione riguarda soltanto il kernel).

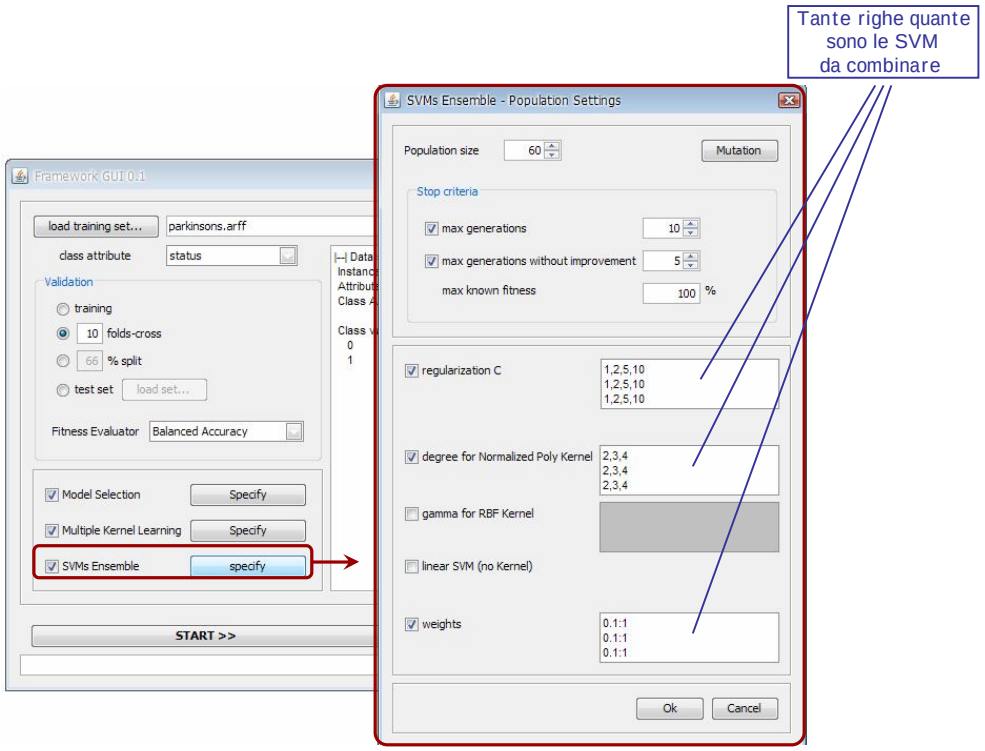

FIGURA 4.8: GAFrameworkGUI 1.0 - Popolazione per la combinazione di SVMs

Infine, il meccanismo di mutazione è un'ulteriore proprietà, comune alle tre popolazioni, che può essere impostata dall'utente attraverso il bottone "Mutation" presente nelle schermate delle 3 popolazioni. In figura 4.9 è mostrata l'interfaccia per la selezione delle impostazioni da utilizzare.

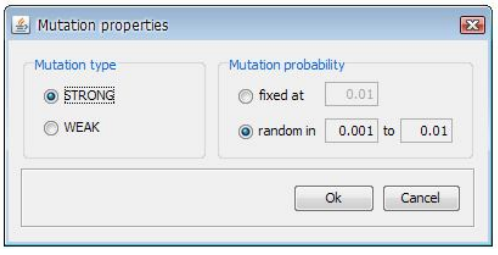

FIGURA 4.9: GAFrameworkGUI 1.0 - Impostazioni di mutazione

Capitolo 5

Altre Meta-euristiche

5.1 Tabu-Search

Le radici di Tabu-Search (TS) risalgono agli anni 70, ma fu presentata per la prima volta nella sua formulazione corrente da Glover (1986) [85]. Rappresenta una tecnica di ricerca locale (*Neighbourhood Search method*), ovvero una procedura iterativa che genera una nuova soluzione j tra quelle nell'intorno $N(i)$ della soluzione corrente i. I metodi di ricerca locale più famosi sono senza dubbio quelli di discesa iterativa, di cui la tecnica TS può essere considerata una versione più elaborata.
Il meccanismo di exploration è potenziato seguendo l'idea che è necessario tenere traccia non solo delle informazioni locali, ma anche di quelle relative al processo esplorativo stesso. A tale scopo TS utilizza una memoria, caratteristica essenziale della tecnica di ricerca: la memoria guida il processo di selezione della nuova soluzione j nell'intorno $N(i)$ della soluzione corrente i, vietando alcune scelte (soluzioni tabu). Visto che la costruzione dell'intorno $N(i)$ varia ad ogni iterazione, la TS ricade tra le tecniche note come metodi dinamici di ricerca locale (*Dynamic Neighbourhood Search methods*).
Un altro aspetto cruciale di TS è l'idea che in determinati momenti della ricerca è utile accettare anche soluzioni che non producono un miglioramento della funzione obiettivo, in modo da non rimanere intrappolati in minimi locali. Questo concetto viene mutuato da un'altra meta-euristica che può ritenersi la progenitrice di TS, denominata *Simulated Annealing* (SA); a differenza di quest'ultima, TS sceglie la migliore soluzione j nell'intorno $N(i)$ mentre SA non guida tale scelta. Il meccanismo di accettazione delle soluzioni "non migliorative" crea tuttavia il rischio di scegliere soluzioni già visitate in precedenza e di generare di conseguenza dei cicli. E' in merito a tale aspetto che l'uso della memoria diventa cruciale, evitando appunto di considerare soluzioni visitate "di recente". La memoria della tecnica TS, generalmente definita *tabu list*, ha una lunghezza prefissata ℓ

e garantirà quindi di non incorrere in cicli di lunghezza ℓ.
La tabu-list costituisce una sorta di "memoria a breve termine" e permette di implementare un meccanismo noto come *intensificazione*: si accettano soluzioni "non migliorative" nell'intorno dell'ottimo corrente. Affinchè la tecnica possa ulteriormente allontanarsi da quello che può risultare un ottimo locale, la tecnica TS prevede l'utilizzo di un'ulteriore memoria, "di medio-lungo termine", introducendo una penalità sul processo di intensificazione: se dopo un certo numero di iterazioni la ricerca non si allontana dall'ottimo corrente interviene un meccanismo noto come *diversificazione*, che consiste nel determinare soluzioni "lontane" dall'ottimo corrente in modo da attivare il meccanismo di exploration sull'intero spazio delle possibili soluzioni.
Intensificazione e diversificazione delle soluzioni si alternano quindi durante la tecnica TS. In figura 5.1 è riportato lo schema di principio della meta-euristica. Inizialmente

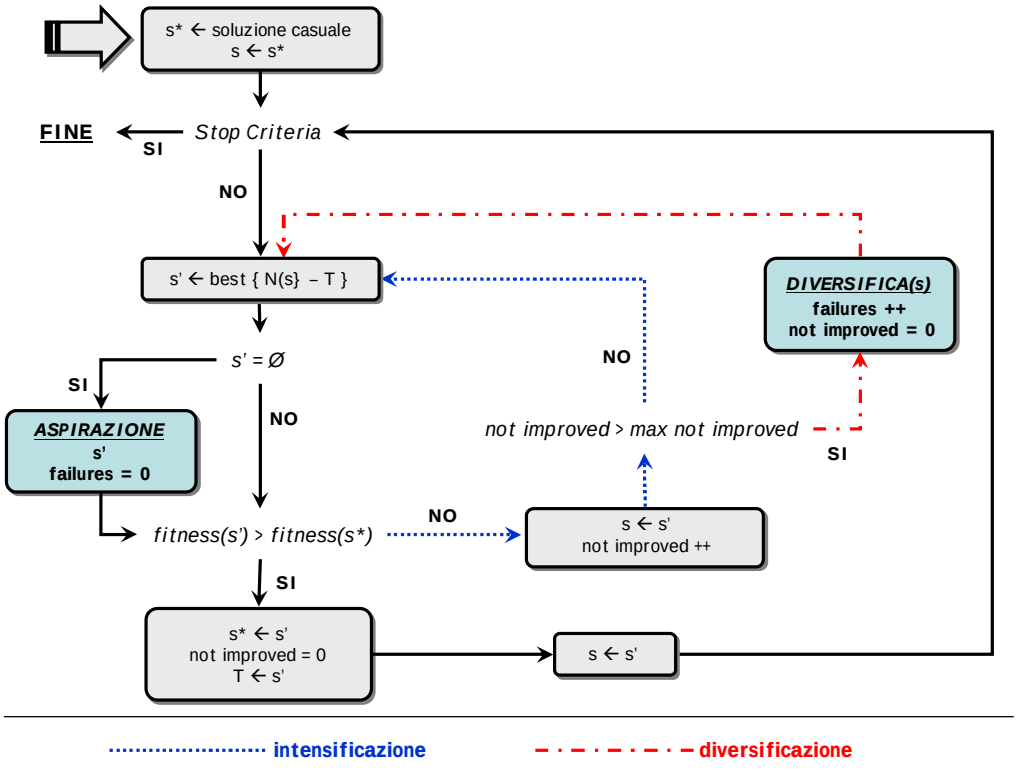

FIGURA 5.1: Schema di principio della Tabu-Search

viene generata in modo casuale una soluzione che costituirà l'ottimo corrente; se un prefissato criterio di terminazione viene soddisfatto (ad esempio si ottiene una soluzione "sufficientemente" buona rispetto al valore di ottimo noto, è stato raggiunto un numero massimo di iterazioni completo o non si è avuto alcun miglioramento dopo un certo numero prefissato di iterazioni) la ricerca termina. In caso negativo si sceglie la migliore

soluzione "non tabu" ottenuta dalla corrente attraverso tutte le possibili mosse (ammissibili). Se tale soluzione non esiste si applica un meccanismo di "aspirazione" (*aspiration*), scegliendo ad esempio la migliore soluzione tra le tabu o una qualsiasi tra esse con probabilità proporzionale alla loro bontà; viene inoltre azzerato il numero di fallimenti della ricerca.
Il valore di funzione obiettivo ottenuto in corrispondenza della soluzione "candidata" (migliore ammissibile non tabu o ottenuta per aspirazione) viene confrontato con l'ottimo corrente: se vi è un miglioramento allora l'ottimo corrente viene aggiornato, si azzera il numero di intensificazioni consecutive senza miglioramento e si inserisce la nuova soluzione nella tabu list; si controlla infine se è stato raggiunto il criterio di terminazione, ed in caso negativo si itera nuovamente. Se, al contrario, la soluzione candidata non produce un valore di funzione obiettivo migliore rispetto a quello corrente si innesca il processo di intensificazione (rappresentato con una linea blu in figura 5.1): si incrementa il numero di intensificazioni consecutive senza miglioramento e si controlla se è stato raggiunto il numero di massimo prefissato. In caso affermativo si innesca il processo di diversificazione (rappresentato con il colore rosso in figura 5.1): si genera una nuova soluzione "distante" dall'ottimo corrente, si incrementa il numero di fallimenti della ricerca e si azzera il numero di intensificazioni consecutive senza miglioramento. In caso negativo, al contrario, si ritorna alla generazione della migliore soluzione non tabu ammissibile, scelta nell'intorno della soluzione candidata.

5.1.1 Stato dell'arte di TS per i task di classificazione SVM

Lebrun *et al* [86] hanno proposto un approccio volto ad unificare in un unico problema di ottimizzazione la feature selection, la semplificazione dell'insieme di addestramento ed il *tuning* dei parametri di un classificatore SVM. L'obiettivo è di produrre *funzioni decisionali binarie* (Binary Decision Function - BDF), ovvero classificatori SVM, che siano efficienti e a bassa complessità.
Il criterio da ottimizzare è stato denominato *Qualità della Funzione Decisionale* (Decision Function Quality - DFQ) e prende in considerazione il tasso di riconoscimento, il numero di support vectors e il numero di features selezionate. In realtà, come specificano gli autori, il criterio rispecchia due aspetti fondamentali di un generico modello decisionale: il tasso di riconoscimento e la complessità. La DFQ q di un modello θ è quindi data da $q(\theta) = R_R(h_0) - C_P(h_0)$, in cui R_R è il tasso di riconoscimento, C_P la complessità e $h0$ la funzione decisionale. Nel caso incui il modello decisionale θ sia un classificatore SVM, la sua complessità dipende sia dal numero di support vectors che dal numero di features selezionate; il valore di complessità di una SVM proposto dagli autori è dato da:

$$C_p(h_0) = c_{p1} \log_2(n_{SV}) + c_{p2} \log_2(\mathbf{cost}(\beta))$$

dove β è un vettore di valori booleani di dimensione n che rappresenta le features selezionate. Sia k_i il costo per l'estrazione della i-esima feature, allora il valore di $\mathbf{cost}(\beta)$ relativo all'insieme di features selezionate è pari a $\mathbf{cost}(\beta) = \sum \beta_i k_i$. Se il costo k_i è sconosciuto può essere fissato a 1 per tutte le features. Le costanti c_{p1} e c_{p2} gestiscono invece il trade-off tra il miglioramento del tasso di classificazione e la riduzione della complessità.

Il criterio DFQ relativo ad un modello θ viene valutato secondo il seguente schema: l'intero insieme di dati S è partizionato in due sottoinsiemi S_t e S_v, rispettivamente insieme di addestramento e di validazione, con $|S_t| = \frac{2}{3}|S|$ e $|S_v| = \frac{1}{3}|S|$; si ottiene la funzione decisionale h_0 tramite addestramento su S_t e la si valutata su S_v. Tasso di riconoscimento R_R e complessità C_P ottenuti sull'insieme di validazione S_v sono infine utilizzati per ottenere il valore q di DFQ. Gli autori sottolineano come nel caso di dataset sbilanciati nelle classi sia più opportuno utilizzare il tasso bilanciato di errore (Balanced Error Rate - BER), misura complementare alla Balanced Classification Accuracy (BCA).

Per quanto concerne l'implementazione della meta-euristica di ricerca, gli autori partono dalla considerazione che il problema è selezionare un modello ottimale θ^* per una certa funzione q con c_{p1} e c_{p2} fissati. Un generico modello θ è rappresentato da un insieme di m variabili intere, $\theta = (\theta_1, \ldots, \theta_m) = (\beta_1, \ldots, \beta_n, k, C', \sigma')$, in cui k rappresenta il livello di semplificazione, β l'insieme di features selezionate, C il parametro di regolazione della formulazione C-SVC e σ il parametro del kernel gaussiano.

Nell'implementazione TS proposta dagli autori, una mossa base corrisponde ad aggiungere una quantità $\delta \in [-1; 1]$ al valore di una componente θ_i del modello, naturalmente preservando il soddisfacimento dei vincoli relativi al modello ($\forall i \in [1, \ldots, m]$, $\theta_i \in [\min(\theta_i); \max(\theta_i)]$). Viene quindi individuata la lista di tutte le soluzioni nell'intorno della corrente raggiungibili attraverso mosse basi ammissibili; tra queste, viene scelta la soluzione non *tabu* con miglior DFQ.

Gli autori indicano una rilevante problematica nella loro implementazione: testare tutte le possibili mosse quando i dati sono descritti da un elevato numero di features può essere particolarmente *time-consuming*. In particolare non è utile, secondo gli autori, andare a testare tutte le mosse sulle features se la soluzione (classificatore SVM) non risulta già sufficientemente promettente; focalizzarsi sulle mosse relative solo ai parametri della SVM o del livello di semplificazione risulta una scelta più efficiente per la scoperta di nuove regioni promettenti.

Viene inoltre implementato anche un meccanismo di diversificazione, in modo da evitare che la TS rimanga intrappolata in ottimi locali: quando il meccanismo di intensificazione fallisce per un certo numero di iterazioni consecutive, vengono effettuate delle mosse con $\delta > 1$, mosse che interessano tutte le componenti della soluzione eccetto quelle relative alle features selezionate (per la problematica precedentemente menzionata).

I risultati indicati dagli autori dimostrano che gli obiettivi da loro prefissati vengono raggiunti: i modelli ottenuti hanno elevate capacità di generalizzazione, hanno bassi livelli di complessità e utilizzano sottoinsiemi efficienti di features. Altra importante conclusione riportata dagli autori riguarda ulteriori risultati sperimentali per un'applicazione di segmentazione di immagini di un microscopio cellulare: l'operazione di *pixel classification* viene realizzata in modo efficace ed efficiente; tale risultato sarà di particolare aiuto nel supporto diagnostico del cancro.

5.1.2 Framework di classificazione ad alto livello con TS

Una versione del framework di classificazione ad alto livello basato su GA e proposto nel presente lavoro di ricerca è stata implementata anche utilizzando la meta-euristica TS. Lo scopo è quello di fornire un confronto tra la bontà delle due differenti tecniche di ricerca delle soluzioni ottimali in modo da apprezzarne i relativi pro e contro.
L'implementazione basata su TS può tuttavia essere confrontata soltanto con quella basata su GA a soli caratteri discreti. L'algoritmo TS implementa lo schema di principio illustrato nella precedente figura 5.1, in modo separato ed indipendente tra i tre task: Model Selection, Multiple Kernel Learning e Sistema di Combinazione tra SVMs.
Le soluzioni ricercate da TS, per le tre problematiche, hanno la stessa forma dei cromosomi definiti nell'approccio basato su GA (figura 4.3). La differenza sostanziale tra le metodologie è che nell'approccio GA un insieme di soluzioni evolve trasmettendo le "migliori caratteristiche" da una generazione all'altra, mentre nella tecnica TS viene sfruttata soltanto l'informazione in un intorno dell'unica soluzione corrente (intensificazione) ed il processo di diversificazione delle soluzioni interviene solo in seguito ad un fallimento del meccanismo di intensificazione.
E' difficile stabilire a priori quale metodologia sia più promettente, e molto dipende dal valore assegnato ai parametri interni di ognuna delle due meta-euristiche, sostanzialmente: la dimensione delle popolazioni e la probabilità di mutazione nell'approccio GA realizzato, la lunghezza della tabu-list ed il momento in cui attivare il processo di diversificazione nell'approccio TS implementato.

5.2 Ant Colony Optimization

Una meta-euristica di ricerca molto interessante è nota con il termine *Ant Colony Optimization* (ACO) ed è nata dall'idea di simulare uno dei fenomeni di organizzazione senza dubbio più affascinanti in natura, quello di una colonia di formiche. Tali insetti sono infatti abilissimi nel trovare in tempi rapidi la strada più breve (soluzione di costo minimo) che mette in comunicazione una fonte di cibo con il loro nido (*nest*). La

metodologia formalizzata a partire dal comportamento delle formiche naturali si presta particolarmente bene, per analogia, a problemi di ottimizzazione discreta, in generale per problemi di ordinamento (*Sequencing Problems*), come i problemi di cammini su grafi. La formulazione della metodologia, infatti, viene generalmente presentata in relazione al ben noto problema del commesso viaggiatore (Traveling Salesman Problem - TSP).
E' infatti possibile immaginare che le formiche si muovano su un grafo con un numero finito di nodi, in cui ogni arco, ovvero ogni cammino tra due nodi adiacenti, ha un certo costo, ad esempio il tempo che una formica impiega a percorrerlo.
Uno dei concetti fondamentali della meta-euristica ACO è lo "stato", ovvero la sequenza ordinata di nodi del grafo visitati da una formica: lo stato rappresenta dunque la memoria della formica. Contemporaneamente, sul grafo, si muovono più formiche (colonia), in modo completamente asincrono. Esse sono inoltre in grado di scambiarsi informazioni sulla bontà dei percorsi depositando una scia di "feromoni" lungo il loro cammino: per eseguire un passo successivo, ogni formica considera sia il peso proprio degli archi percorribili (valori euristici) e sia la traccia di feromoni lasciata sugli stessi archi dalle altre formiche durante la loro esplorazione. In termini implementativi, le formiche usano la cosiddetta *Ant Routing Table*, funzione, in ogni nodo, delle scie di feromoni, dei valori euristici e del cammino già percorso. Una buona soluzione è trovata solo grazie ai risultati collettivi di tutte le formiche.
Per quanto concerne la deposizione dei feromoni esistono due procedure di aggiornamento possibili: l'*aggiornamento attivo passo per passo*, che prevede la deposizione dei feromoni mentre la formica esplora il grafo, e l'*aggiornamento attivo ritardato*, che prevede al contrario che la formica depositi i feromoni solo dopo aver trovato e valutato una soluzione, percorrendo a ritroso il proprio cammino.
Sono inoltre previsti due ulteriori meccanismi agenti sulle scie di feromoni: l'*evaporazione* e l'*azione di folletti*. Il primo deve essere sempre implementato nella meta-euristica e consiste nel far decrescere l'intensità delle scie di feromoni con il passare del tempo. Questo meccanismo rappresenta l'exploiration della tecnica, evitando che si converga troppo velocemente verso un ottimo locale e favorendo al contempo l'esplorazione di nuove regioni nello spazio delle soluzioni ammissibili.
L'azione dei folletti rafforza al contrario il meccanismo di exploitation (modifica delle intensità delle tracce di feromoni): i folletti possono depositare o rimuovere, a seconda delle implementazioni, feromoni, in modo da condizionare il processo di ricerca delle formiche. La soluzione più comune è la deposizione di feromoni extra su un percorso particolarmente "buono" trovato da una formica (premiazione); in genere l'azione dei folletti è più formalmente nota con il termine di *aggiornamento passivo* di feromoni.

5.2.1 Dai Sequencing ai Subset Problems

La meta-euristica ACO risulta particolarmente adatta alla risoluzione di problematiche di ordinamento (*Sequencing Problems*), tuttavia è stata proposta una variante che permette di utilizzare l'approccio anche in corrispondenza di problemi di selezione di sottoinsiemi (*Subsets Problems*) [87]

Al fine di comprendere come la metodologia ACO possa essere trasposta dai Sequencing ai Subsets Problems, si parte generalmente dalla sua applicazione al TSP per poi specificare le differenze e quindi le dovute variazioni per la sua applicazione a problematiche di selezione quali il *Multiple Knapsack Problem* (MKP).

Date n città ed un insieme di distanze tra di esse, il problema (di ordinamento) del commesso viaggiatore (TSP) ha come obiettivo trovare il cammino chiuso di costo minimo che visiti esattamente una volta ogni città. In termini formali:

$$\min COST(i_1, \ldots, i_n) = \sum_{j=1}^{n} d(Ci_j, Ci_{j+1}) + d(Ci_n, Ci_1) \tag{5.1}$$

con $d(C_x, C_y)$ la distanza tra la città x e la città y.
Sia $b_i(t)$, con $i = 1, \ldots, n$ sia il numero di formiche nella città i al tempo t e sia $a = \sum_{i=1}^{n} b_i(t)$ il numero totale di formiche al tempo t. Sia inoltre $\tau_{ij}(t+n)$ l'intensità della traccia di feromoni sulla connessione (i, j) al tempo $t+n$, data da:

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t, t+n); \tag{5.2}$$

con il coefficiente $0 < \rho \leq 1$ che rappresenta il tasso di evaporazione.
$\Delta\tau_{ij}(t, t+n) = \sum_{k=1}^{a} \Delta\tau_{ij}^{k}(t, t+n)$, con $\Delta\tau_{ij}^{k}(t, t+n)$, è la quantità di feromoni, per unità di lunghezza, lasciata sulla connessione (i, j) dalla k-esima formica al tempo $t+n$, ed è data dalla seguente formula:

$$\Delta\tau_{ij}^{k}(t+n) = \begin{cases} \frac{Q}{L_k} & se\ la\ k-esima\ formica\ usa\ la\ connessione\ (i,j) \\ 0 & altrimenti \end{cases}$$

dove Q è una costante e L_k è la lunghezza del tour trovato dalla k-esima formica. L'intensità della scia di feromoni su ogni arco al tempo zero ($\tau_i j(0)$) è impostata ad un valore molto piccolo.
Nel costruire il proprio percorso ogni formica sceglie un nuovo arco secondo il costo dell'arco stesso ed il vantaggio ottenuto dall'inserirlo nel cammino, vantaggio espresso dalla scia di feromoni lasciata dalle altre formiche durante la loro esplorazione. La funzione di probabilità secondo la quale la k-esima formica nella città i sceglie di visitare la città j

è data da:

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in allowed_k(t)}[\tau_{ih}(t)]^{\alpha}[\eta_{ih}]^{\beta}} & se \;\; j \in \; allowed_k(t) \\ 0 & altrimenti \end{cases}$$

dove $allowed_k(t)$ è l'insieme delle città non visitate dalla k-esima formica al tempo t e η_{ij} rappresenta un valore euristico locale. Nel caso specifico del TSP $\eta_{i,j} = \frac{1}{d(C_i,C_j)}$, anche detta visibilità.
I parametri α e β controllano il trade-off tra la traccia di feromoni e la visibilità, ovvero l'importanza relativa dei due fattori nella scelta della prossima città da visitare. In termini semplicistici, le "strade" più brevi e più "trafficate" sono quelle preferite nella scelta.

I problemi di Subsets sono piuttosto differenti da quelli di Sequencing: in termini generali è necessario individuare, a partire da un insieme S di s elementi, il miglior sottoinsieme di n elementi, soddisfacendo determinati vincoli; non compare in alcun modo il concetto di cammino e quindi di ordinamento degli elementi. Inoltre, nei problemi di Sequencing le soluzioni hanno una lunghezza fissa, implicando che l'aggiornamento delle tracce di feromoni può essere effettuato ogni p passi, ovvero quando tutte le formiche hanno completato la costruzione della propria soluzione di ordinamento. Le soluzioni dei problemi di Subset, al contrario, non hanno una lunghezza fissa ed è perciò richiesto impostare a priori un numero N_{max} al fine di determinare la fine del ciclo di costruzione da parte delle formiche. La meta-euristica ACO va quindi modificata di conseguenza.
Innanzitutto, i feromoni vengono ora depositati sugli elementi dell'insieme S con il significato che gli elementi con una più forte traccia di feromoni sono più vantaggiosi. L'intensità della traccia di feromoni sull'elemento i al tempo $t + N_{max}$ è data da:

$$\tau_i(t + N_{max}) = (1 - \rho)\tau_i(t) + \Delta\tau_i(t, t + N_{max}) \tag{5.3}$$

con $N_{max} < n$ il numero massimo di elementi che ad un formica è permesso selezionare per la costruzione della soluzione.
Una delle prime formulazioni di ACO per i problemi di Subset è relativa al Multiple Knapsack Problem (MKP). Il problema può essere formulato come:

$$\begin{aligned} &maximize \;\; \textstyle\sum_{j=1}^{n} p_j x_j \\ &subject \;\; to \;\; \textstyle\sum_{j=1}^{n} r_{ij} x_j \leq c_i \;\; i = 1, \ldots, m \\ &x_j \in 0, 1 \;\; j = 1, \ldots, n \end{aligned}$$

Sia $I = 1, \ldots, m$ e $J = 1, \ldots, n$ con $c_i \geq 0$ per ogni $i \in I$. Un MKP ben formulato assume che $p_j > 0$ e $r_{ij} \leq c_i \leq \sum_{j=1}^{n} r_{ij}$ per ogni $i \in I, j \in J$, visto che qualsiasi violazione di tali condizioni farebbe sì che qualche x_j venga fissata a zero o alcuni vincoli vengano eliminati.

Per risolvere MKP le formiche cercano un sottoinsieme di n elementi in grado di massimizzare il profitto (funzione obiettivo) e soddisfare tutti i vincoli relativi alle risorse.

5.2.2 Stato dell'arte di ACO per i task di classificazione SVM

L'applicazione della meta-euristica ACO nei task di classificazione è rivolta generalmente all'attività di Feature Selection. Un interessante e recente lavoro (2008) è quello di Xiong e Wang [88], che realizza un sistema ibrido per la Feature Selection basato sulla combinazione dell'approccio ACO e della tecnica di apprendimento SVM. Ogni volta che una formica seleziona un sottoinsieme di features, la bontà della scelta viene valutata attraverso le prestazioni offerte da un classificatore SVM addestrato su quello specifico insieme di features selezionate.
I valori euristici vengono inizializzati attraverso una misura della correlazione tra i valori di ogni feature e la classe. Gli autori partono dalla misura *F-score* proposta da Chen e Lin [89], ma questa ha il difetto di poter essere utilizzata soltanto in corrispondenza di problemi di classificazione binaria. Xiong e Wang propongono quindi una generalizzazione di F-score ad m classi, denominata *F-merits*, che presenta la seguente espressione:

$$F-merits(i) = \frac{max(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_m) - min(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_m)}{\sqrt{\sum_{j=1}^{m} \frac{1}{c_j} \frac{1}{c_j - 1} \sum_{k=1}^{c_j} (x_k - \overline{X_j})^2}}$$

dove il numeratore è la differenza tra il massimo ed il minimo valore medio della porzione della feature i appartenente alle m classi, il denominatore è la deviazione standard media e c_j è il numero di elementi nella porzione di classe j:

$$c_j = \left|\left\{x | x_{\{class=j\}}\right\}\right|, \quad \overline{X}_j = \frac{\sum_{k=1}^{c_j}}{c_j}, \quad x_k \in \left\{x | x_{\{class=j\}}\right\}$$

Al fine di creare una "mappa" delle distanze tra i nodi del grafo, gli autori propongono di mappare ogni feature in un nodo e calcolare la distanza tra due nodi come:

$$Distance(i,j) = \begin{cases} \frac{1}{F-merits(i)} + \frac{1}{F-merits(j)} & se \;\; i \neq j \\ 0 & altrimenti \end{cases}$$

Inoltre gli autori normalizzano tale distanza in modo da avere un valore compreso tra 0 e 1.

$$Distance(i,j)_{normal} = \frac{D(i,j) - \min(D(i,j))}{\max(D(i,j)) - \min(D(i,j))}$$

Per quanto concerne l'evaporazione, il valore viene determinato in modo dinamico ed in base alla velocità di convergenza e alla capacità di ricerca della colonia, in modo da non

rimanere intrappolati in un ottimo locale. La proposta degli autori è la seguente:

$$Ph(i,j) = (1-\rho)Ph(i,j) + \rho\frac{\gamma Ph(0)}{1-accuracy}$$

$$\rho = 0.95\rho, \quad \gamma = 0.95\gamma$$

dove ρ è il fattore di evaporazione mentre γ regola il feedback positivo nell'incremento dei feromoni, infine $Ph(0)$ è l'intensità dei feromoni.
La probabilità di transizione, basata su valori euristici e feromoni, adottata dagli autori è la seguente:

$$P^k(i,j,t) = \frac{[Ph(i,j,t)]^\alpha \left[\frac{1}{D(i,j,t)}\right]^\beta}{\sum_{s\in unselected(k)}\left([Ph(i,s,t)]^\alpha \left[\frac{1}{D(i,s,t)}\right]^\beta\right)}$$

Dopo aver calcolato le distanze normalizzate tra le features (valori euristici), l'algoritmo proposto dagli autori prevede che ogni formica scelga in modo puramente casuale il numero di features da selezionare ed una feature iniziale. In seguito, sulla base della probabilità di transizione, vengono scelte, una per volta, le altre features, fino al raggiungimento del massimo numero previsto. Una volta che la formica ha terminato la sua selezione, un classificatore SVM viene addestrato sull'insieme di features selezionate per misurarne la bontà (accuratezza), aggiornando gli ottimi correnti, locale (all'interno della colonia al tempo t) e globale (fino al tempo corrente t). Quando l'intera colonia ha visto valutate le proprie scelte si passa all'aggiornamento dei feromoni; inoltre, se l'ottimo corrente (accuratezza) rimane costante per tre iterazioni consecutive si attiva il processo dinamico di aggiornamento dell'evaporazione. Infine, gli autori hanno utilizzato $\alpha = 2$ e $\beta = 1$ in accordo alla loro esperienza sperimentale.

Un'applicazione pratica dell'approccio ibrido $ACO + SVM$ è stata proposta per la scoperta di biomarkers [90]. Tale ricerca viene eseguita attraverso l'analisi spettrale di masse e richiede una lunga fase di preparazione del dato in cui deve essere posta particolare attenzione. Dopo il preprocessamento del dato grezzo, finalizzato in genere alla rimozione di artefatti, segue una fase definita *Peak Screening* per l'individuazione e l'eliminazione di picchi associati a fattori noti (come ad esempio l'età ed il genere). Al termine di questa fase segue il processo di identificazione dei picchi associati alla patologia sotto esame (*Peak Selection*). Gli autori propongono un sistema ibrido $ACO + SVM$ per la selezione dei picchi dallo spettro ottenuto con tecnica *Matrix-Assisted Laser Desorption/Ionization Time-Of-Llight (MALDI-TOL)*. L'idea è del tutto simile a quella definita da Xiong e Wang, ma non viene chiarito se venga utilizzato l'approccio ACO per problemi di Sequencing (come in Xiong e Wang) o piuttosto la formulazione per problemi

di Subsets.
I risultati ottenuti dagli autori indicano che il ristretto insieme di picchi, selezionato dall'approccio $ACO + SVM$, fornisce una classificazione altamente sensibile e specifica tra pazienti affetti da cirrosi e pazienti affetti da carcinoma epatocellulare (*hepatocellular carcinoma - HCC*).

Entrambi i lavori presentati riguardano espressamente la Feature Selection, ma non trattano in alcun modo la tematica di Model Selection. Il classificatore SVM utilizzato negli approcci ibridi proposti è difatti fissato a priori ed utilizzato esclusivamente per misurare l'affidabilità del sottoinsieme di feature selezionate. E' logico pensare che gli autori abbiano in realtà testato differenti configurazioni del classificatore SVM, usando un semplice approccio a griglia (anche se ciò non è espressamente specificato).
Un approccio più interessante è invece presentato nel lavoro di Acevedo *et al* [91], in merito a problemi di riconoscimento olfattivo elettronico (*Electronic Nose*). Gli autori implementano la Model Selection per una SVM a kernel RBF tramite tecnica ACO, in particolare utilizzano un kernel RBF multi-γ (un parametro per ogni features) implementando in tal modo la feature selection in modalità *wrapper* (in modo analogo alla definizione che Chapelle et al hanno utilizzato per la loro Model Selection, sezione 2.3).

5.2.3 Approccio proposto per il Multiple Kernel Learning con ACO

Per le finalità della ricerca anche la metodologia ACO è stata considerata per una possibile implementazione del Framework, seppure la tecnica non si presti ad una immediata applicazione né sembri semplice intravedere possibili modifiche per adattarla agli scopi della ricerca stessa.
Qui si vuol tuttavia presentare uno spunto ritenuto sufficientemente promettente ed interessante e che riguarda l'implementazione del solo task di *Multiple Kernel Learning* tramite la formulazione ACO per Subsets Problems. L'idea iniziale consiste nel determinare un insieme di kernel basali, differenti per tipologia o per valore dei parametri, e fare in modo che le formiche della colonia ne individuino un sottoinsieme la cui combinazione vada ad ottimizzare le prestazioni di un classificatore SVM. Il problema così formulato risulta molto simile all'applicazione dell'ACO per la Feature Selection, dove l'insieme delle features iniziali è, in questo caso, sostituito dall'insieme dei kernel basali.
In figura 5.2 è illustrato lo scenario: tutti i kernel che costituiscono l'insieme basale (nel caso in figura sono N kernel polinomiali normalizzati e M kernel RBF) vengono inizialmente usati singolarmente per addestrare e valutare i classificatori SVM, in modo da ottenere il valore euristico associato ad ogni kernel, ovvero il vantaggio che si otterrebbe nell'adottare soltanto il singolo kernel. In analogia con la Feature Selection basata su ACO, questo valore euristico può essere assimilato al livello di correlazione di ogni

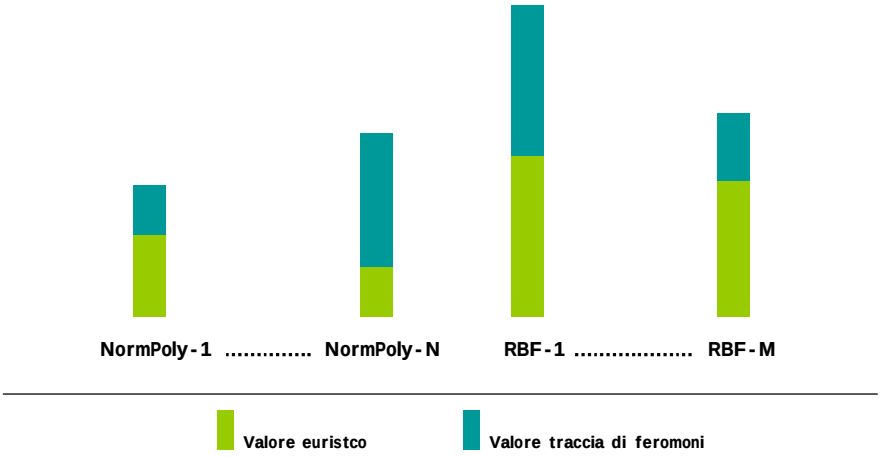

FIGURA 5.2: ACO per selezione da un insieme di kernel basali

feature verso il valore di classe (informazione univariata): nel caso dei kernel è il valore di affidabilità (ad esempio accuratezza, sensibilità o qualsivoglia indice di prestazione) riscontrato da un classificatore SVM con quella specifica configurazione kernel.

L'idea è stata ulteriormente complicata nell'ottica di voler ottimizzare anche i coefficienti della combinazione lineare di kernel basali e non di limitarsi alla mera selezione. Questo nuovo scenario è illustrato in figura 5.3 in cui si può osservare un'ulteriore "dimensione" lungo cui le formiche possono spostarsi per effettuare la loro scelta, ovvero un certo numero di valori prefissati per i coefficienti della combinazione kernel.

I valori euristici sono calcolati come specificato per l'idea base, quindi sono costanti a prescindere dal coefficiente scelto e dipendono solo ed esclusivamente dalla configurazione kernel.

Nella seguente figura 5.4 è schematizzato l'algoritmo ACO implementato per il Multiple Kernel Learning. Il primo passo è il calcolo dei valori euristici, ovvero la valutazione di tutti i classificatori SVM che hanno per kernel quelli basali presi singolarmente. Successivamente ogni formica seleziona N coppie $(coefficiente, configurazionekernel)$, con N fissato, basando la scelta sui valori euristici e sulle tracce di feromoni lasciate in corrispondenza di ogni coppia (inizialmente le intensità delle tracce di feromoni sono molto basse); la scelta di ogni formica viene quindi memorizzata.

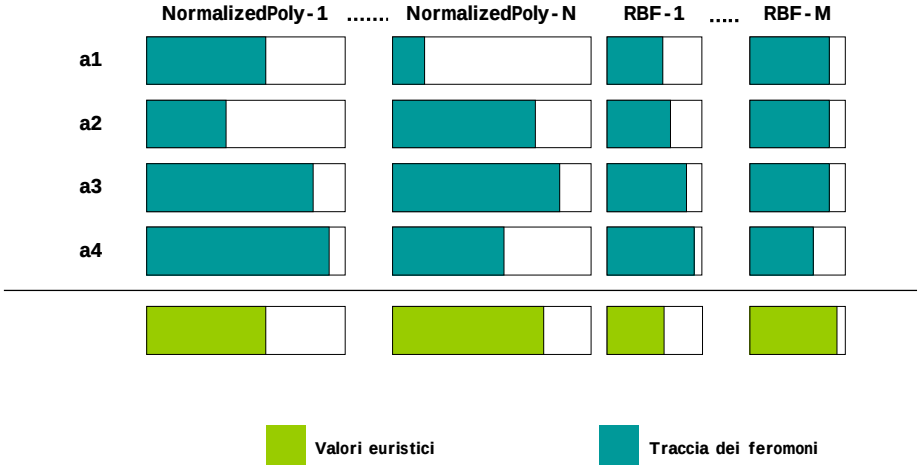

FIGURA 5.3: ACO per Multiple Kernel Learning

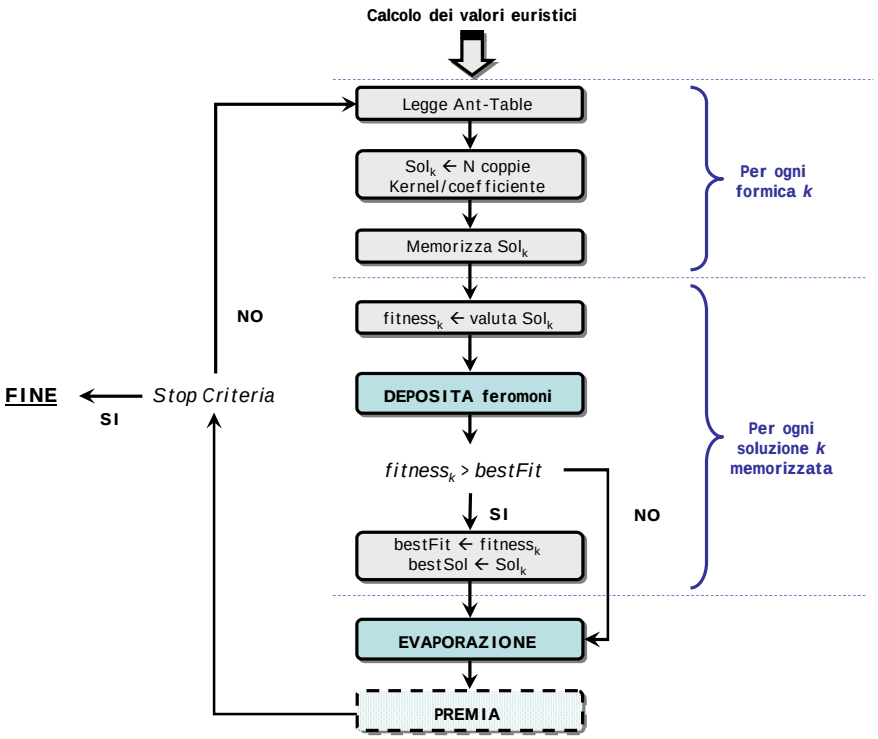

FIGURA 5.4: Schema di principio di ACO per il Multiple Kernel Learning

Quando tutte le formiche della colonia hanno effettuato la loro scelta ogni combinazione kernel viene testata e viene depositata una quantità di feromoni proporzionale all'affidabilità riscontrata, divisa equamente sulle N coppie $(coefficiente, configurazionekernel)$ (figura 5.3). Inoltre, se il livello di bontà della soluzione individuata è migliore dell'ottimo corrente, questo viene aggiornato.
Passo successivo è l'evaporazione delle tracce di feromoni ed un'eventuale operazione di "premiazione" dell'ottimo corrente (operazione definita anche "azione dei folletti", come specificato nella sezione 5.2 "Ant Colony Optimization").
Se infine il criterio di terminazione è stato soddisfatto allora l'algoritmo termina, altrimenti la colonia ricomincia il suo processo esplorativo.

In merito alla specifica implementazione realizzata, devono essere specificate alcune variazioni che si sono rese necessarie alla luce dell'esperienza sperimentale. Affinché la colonia non converga troppo velocemente e non rimanga intrappolata in ottimi locali, la politica di deposizione dei feromoni è stata modificata: dopo aver valutato la scelta della k-esima formica, l'intensità dei feromoni sulle coppie $(coefficiente, configurazionekernel)$ da essa scelte non è incrementata ma aggiornata nel seguente modo: se l'intensità calcolata sulla base dell'affidabilità del classificatore è maggiore dell'intensità di feromoni già presente sulla coppia, allora viene aggiornata con il nuovo valore, altrimenti la formica non deposita alcun feromone.
Questo tipo di politica serve ad evitare che le coppie frequentemente selezionate incrementino eccessivamente l'intensità della propria traccia di feromoni, anche se magari presentano in realtà un basso livello di affidabilità individuale.
Inoltre, questo tipo di politica permette di aggiornare l'intensità della traccia di feromoni sulle coppie in relazione alla bontà registrata per l'intera soluzione (N coppie scelte dalla singola formica). Un esempio con soluzioni costituite dalla combinazione di 3 kernel:
si supponga che sulle coppie C_1, C_2, C_3 vi sia già depositata una traccia di feromoni pari a 0.25 e che sulle coppie C_4, C_5, C_6 vi sia invece un'intensità pari a 0.18. Si supponga ora che una nuova formica abbia selezionato le coppie C_1, C_4 e C_6, ottenendo un'affidabilità pari a 0.60, ovvero 0.20 feromoni da depositare per ogni coppia della soluzione. Secondo la politica precedentemente descritta, la traccia su C_1 non viene aggiornata ($0.20<0.25$), contrariamente alla C_1 e la C_6: va fatto notare che in questo modo viene inoltre mantenuta inalterata l'intensità dei feromoni della scelta $(C_1; C_2; C_3)$.
Infine, il meccanismo di selezione tiene anche conto del fatto che, al tempo t, possono esserci coppie mai selezionate; per ovviare a ciò la probabilità di selezione dipende anche dal fatto che una coppia sia stata o meno selezionata durante il processo di ricerca, questo accorgimento permette di migliorare il meccanismo di *exploration* delle soluzioni.

Capitolo 6

Esperimenti e risultati

In questo capitolo sono riportati i risultati della sperimentazione del Framework di classificazione ad alto livello basato su GA che è stato proposto nell'ambito del presente lavoro di ricerca. In congiunzione vengono riportati anche i risultati ottenuti per la realizzazione dello stesso Framework con TS e, laddove ritenuto utile, anche i risultati ottenuti con il classico approccio a griglia.
Inoltre vengono sintetizzati i risultati di una fase sperimentale del Multiple Kernel Learning basato su metodologia ACO, mutuato e specializzato a partire dalle applicazioni di ACO per la Feature Selection.

I dataset utilizzati per la sperimenazione sono divisi in due tipologie: pubblici e costituiti per scopi di ricerca; in totale si è fatto uso di 8 dataset, tutti relativi a problematiche di classificazione; maggiori dettagli in merito ai problemi e alle caratteristiche dei dataset sono forniti nelle successive sezioni.
Anche i risultati sono essenzialmente distinti coerentemente con le due tipologie di dataset. Tale suddivisione è principalmente dovuta ad una preliminare analisi delle prestazioni del Framework sui dataset pubblici (già noti nell'ambito della ricerca e dello sviluppo di metodologie di classificazione), mentre l'applicazione della tecnica ai dataset "privati" ha permesso di stimare i reali vantaggi offerti dall'approccio proposto rispetto alle sperimentazioni di tipo classico, già precedentemente eseguite sui dataset in merito alle attività di ricerca del Laboratorio di Ingegneria delle Decisioni per i Servizi Sanitari.

Tutte le sperimentazioni sono state eseguite su un laptop Acer TravelMate 7730G: Intel Core 2 Duo P8400 (2.26 GHz, 1066 MHz FSB, 3 MB L2 cache), 4 GB DDR2, 640 GB dual HHD (320 GB x 2), sistema operativo Windows XP, e lanciate direttamente sotto eclipse 3.4.1, su piattaforma Java jdk 1.6.0_11.

6.1 Dataset pubblici

I 5 dataset pubblici, di cui 4 provengono dal database UCI (www.ics.uci.edu/~mlearn/MLRepository.html), sono tutti relativi a problemi di classificazione binaria e sono rispettivamente:

- **WEATHER** - un semplice dataset, completamente fittizio, utilizzato per scopi didattici. Il task di classificazione consiste nello stabilire se una partita di un qualche sport non specificato verrà giocata o meno a seconda delle condizioni meteorologiche, misurate attraverso umidità, temperatura, vento e tipo di giornata (serena, coperta o piovosa).
- **SPECTF** - un dataset proveniente dal dominio della cardiologia. I dati sono relativi al processo di diagnosi basato su immagini SPECT (*Single Proton Emission Computed Tomography*). Per ogni istanza (paziente) si ha diagnosi *normale* o *anormale*.
- **PARKINSONS** - un altro dataset medico-clinico riguardante la patologia Parkinsons. Il problema di classificazione consiste nel distinguere i soggetti sani da quelli affetti da Parkinsons, sulla sola base dei valori assunti da alcuni parametri medico-clinici.
- **WBCP** - una porzione del famoso dataset sul cancro alla mammella (*Winsconsin Breast Cancer*). Il dataset riguarda nello specifico la problematica della prognosi del tumore: ogni istanza rappresenta i dati di follow-up di un paziente che ha manifestato un tumore invasivo ma nessuna evidenza di metastasi distanti nel momento della diagnosi. Ben 30 features sono calcolate dalle immagini digitalizzate dell'esame agoaspirato (*Fine Needle Aspirate* - FNA), un'altra feature memorizza la dimensione del tumore in centimetri mentre un'ultima fornisce infine il numero di linfonodi ascellari positivi al momento dell'intervento.
- **WBCD** - la seconda e più corposa porzione del dataset Winsconsin Breast Cancer, relativa all'aspetto diagnostico della patologia: 30 features sono le misure rilevate dalle immagini digitalizzate dell'esame agoaspirato (*Fine Needle Aspirate* - FNA), lo stesso tipo di misure rilevate per il dataset WBCP.

Tutti i dataset sono stati sottoposti ad una fase di pre-processamento consistente in una procedura di normalizzazione (*scaling* del range di ogni feature nell'intervallo [0;1]). Come già affermato anche in relazione ad altri lavori illustrati nelle sezioni sullo stato dell'arte, tale procedura di pre-processamento evita che le features con range di variabilità più ampi dominino su quelle con range più ristretti e permette in genere di migliorare le

prestazioni dei classificatori SVM, sia in termini di accuratezza che di tempi di calcolo. Solo per il dataset denominato WEATHER si è deciso di riportare i risultati ottenuti con e senza normalizzazione; l'esiguo numero di casi ha infatti permesso di considerare anche la versione non normalizzata senza incrementare eccessivamente i tempi di addestramento dei classificatori.
Nella seguente tabella 6.1 sono riportate le caratteristiche principali dei singoli dataset pubblici:

Nome	Istanze	Istanze$^+$	Istanze$^-$	Features	Features continue	Features intere	Features nominali
WEATHER	14	9	5	4	2	0	2
SPECTF	80	40	40	44	44	0	0
PARKINSONS	195	48	147	22	22	0	0
WBCP	198	47	151	32	32	0	0
WBCD	569	212	357	30	30	0	0

TABELLA 6.1: Principali caratteristiche dei dataset pubblici

6.2 Dataset da attività di ricerca

I 3 dataset costruiti durante attività di ricerca presso il Laboratorio di Ingegneria delle Decisioni per i Servizi Sanitari sono tutti relativi a problemi reali di classificazione binaria nel dominio medico-clinico e sono stati rispettivamente denominati:

- **Noduli Tiroidei** - un dataset relativo a pazienti sottoposti a biopsia di noduli tiroidei, mediante la quale è stata rilevata l'espressione genetica di 6 geni presenti sui tessuti e ritenuti in letteratura candidati per la diagnosi precoce. Ogni soggetto ha riportato una tra due possibili diagnosi: positiva (nodulo maligno) o negativa (nodulo benigno);

- **Musica ed Emozioni** - un dataset costituito durante il progetto di ricerca MIMERICA (http://www.mimerica.it/portal/portal/MIMERICAPortal), più in particolare per la realizzazione di un sistema integrato di supporto alle decisioni mediche nel dominio della terapia per pazienti con gravi cerebrolesioni. Il dataset codifica una serie di parametri relativi all'attività cardiaca di soggetti non cerebrolesi e senza patologie cardiache, calcolati mediante *Heart Rate Variability Analysis* e rilevati durante l'ascolto di 4 differenti brani musicali in grado di indurre specifici stati emotivi. Il task di classificazione si pone l'obiettivo di identificare una possibile relazione tra l'attività cardiaca ed il possibile stato emotivo di un soggetto: positivo (relax, calma, tranquillità, serenità, gioia) o negativo (ansia, tensione,

paura, tristezza, terrore) [92–94]. Il punto cruciale dello studio è che lo stimolo musicale è standard, ripetibile e, in un certo senso, misurabile: una volta indotto il soggetto in uno specifico stato (2 brani musicali sono in grado di indurre emozioni positive e 2 emozioni negative) può essere ricercata la relazione che lega attività cardiaca a stato emotivo. L'obiettivo terapeutico è ottenere un criterio in grado di oggettivare lo stato di tensione/relax per pazienti incapaci di fornire risposte, come pazienti in stato vegetativo o di minima coscienza: è generalmente riconosciuto che solo nei momenti in cui tali pazienti sono in uno stato non tensivo è più opportuno sottoporre stimoli (uditivi, visivi e/o tattili) finalizzati a creare un "contatto" o ottenere una qualche forma di risposta;

- **EDDC** - un dataset costituito durante le attività in merito al progetto di ricerca HEARTFAID (http://www.heartfaid.org/) finanziato dalla commissione europea nel contesto FP6 e rivolto allo sviluppo di una innovativa piattaforma di servizi, tecnologica e distribuita, per una migliore e più efficace gestione dei pazienti affetti da scompenso cardiaco cronico. Il dataset in particolare è relativo ad un gruppo di circa 50 soggetti a cui, ogni 2 settimane, sono stati rilevati alcuni parametri vitali ed il relativo stato di salute (stabilità o nuova destabilizzazione). Dopo una preparazione dei dati, la versione del dataset utilizzata codifica 301 casi (visite) in cui le features sono i valori di 4 parametri vitali e la classe è lo stato di salute alla visita successiva. Le 301 visite selezionate hanno la particolarità di essere relative a pazienti in condizione stabile: il task di classificazione ha l'obiettivo di prevedere il rischio di una nuova destabilizzazione (o di mantenere la stabilità) in 2 settimane [95].

  Deve infatti essere puntualizzato che il riconoscimento precoce di eventuali destabilizzazioni (*Early Detection of Decompensation Conditions - EDDC*) è una delle principali problematiche indirizzate nel dominio della gestione del paziente affetto da scompenso cronico; una previsione o una pronta valutazione permettono di prevenire o ridurre i danni di una nuova destabilizzazione, migliorando la qualità di vita del paziente e riducendo al contempo i costi relativi ad ulteriori riospedalizzazioni. A tal proposito i parametri vitali considerati presentano la proprietà di essere facilmente rilevabili anche direttamente a casa del paziente (*home-care setting*), in modo automatico o semi-automatico.

Anche questi 3 dataset sono stati sottoposti ad una fase di pre-processamento consistente in una procedura di normalizzazione.
Nella seguente tabella 6.2 sono riportate le caratteristiche principali dei singoli dataset "privati":

Nome	Istanze	Istanze$^+$	Istanze$^-$	Features	Features continue	Features intere	Features nominali
Noduli Tiroidei	99	43	56	6	6	0	0
Musica ed Emozioni	104	45	59	24	23	0	1
EDDC	301	11	290	4	4	0	0

TABELLA 6.2: Principali caratteristiche dei dataset privati

6.3 Risultati - Dataset pubblici

6.3.1 Analisi preliminare: Model Selection

Si è ritenuto utile testare inizialmente le prestazioni delle implementazioni delle diverse meta-euristiche in merito al solo problema di Model Selection e sui soli dataset pubblici. L'obiettivo è confrontare i valori di fitness ottenuti da una semplice sperimentazione a griglia con quelli ottenuti dall'applicazione delle meta-euristiche di ricerca (GA a caratteri discreti e TS), permettendo di stabilire se queste ultime siano effettivamente in grado di (ri)trovare gli stessi risultati ottenuti dalla classica sperimentazione a griglia e considerando inoltre il risparmio in termini di tempi di calcolo. Il fitness è stato definito come accuratezza bilanciata (BCA, Balanced Classification Accuracy 3.3) valutata su 10fold-cross validation.
Sono stati previsti i segueti valori per ogni parametro delle configurazioni SVM:

- regolazione C (4 valori): 1, 2, 5, 10;
- tipi di kernel (3 valori): nessun kernel (SVM lineare), RBF e Polinomiale (normalizzato);
- degree per il kernel polinomiale (4 valori): 2, 3, 4, 5;
- γ per il kernel RBF (25 valori): 0.001, 0.002, 0.005, 0.007, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000;

per un totale di $|C| \times (|\gamma| + |degree| + 1) = 4 \times (25 + 4 + 1) = 120$ configurazioni.

Per l'approccio GA la dimensione della popolazione è fissata a 10 individui, la probabilità di mutazione viene scelta casualmente nel range [0.001; 0.01] e la terminazione avviene per una delle seguenti condizioni:

- ottimalità raggiunta (BCA=100%),
- nessun miglioramento del fitness per 10 generazioni consecutive,

- omogeneità della popolazione,
- numero massimo di 100 generazioni raggiunto.

Per l'implementazione TS è stata fissata a 10 la lunghezza della memoria (*tabu list*), a 5 il numero di fasi di *intensificazione* senza alcun miglioramento e a 5 il numero di fallimenti consecutivi della fase di *diversificazione*.
La terminazione per l'implementazione TS avviene se è soddisfatto uno dei seguenti criteri:

- ottimalità raggiunta (BCA=100%),
- numero massimo di 30 iterazioni complete raggiunto.

In tabella 6.3 sono riportati i valori di fitness ottenuti con sperimentazione a griglia, approccio con algoritmi genetici (versione a caratteri discreti) e Tabu-Search.

	Grid-Model Selection		**GA-Model Selection**		**TS-Model Selection**	
	Fitness	**Tempo**	**Fitness**	**Tempo**	**Fitness**	**Tempo**
WEATHER	94.44%	0:00:26	94.44%	***0:00:11***	94.44%	0:00:17
WEATHER (no norm.)	50.00%	0:00:26	50.00%	***0:00:11***	50.00%	0:00:21
SPECTF	80.00%	0:00:34	80.00%	***0:00:17***	80.00%	0:00:22
PARKINSONS	93.09%	0:00:54	93.09%	0:00:37	93.09%	***0:00:35***
WBCP	***70.42%***	0:01:04	64.77%	0:00:24	68.37%	0:00:36
WBCD	***97.55%***	0:04:35	97.46%	0:00:57	***97.55%***	***0:01:45***

Tabella 6.3: Model Selection con: approccio a griglia, GA (a caratteri discreti) e TS

In 4 casi, dei 6 totali, il valore ottimale identificato dalla sperimentazione a griglia viene ritrovato sia attraverso l'approccio GA (a caratteri discreti) che attraverso l'approccio basato su TS. Per questi 4 casi è particolarmente interessante osservare i tempi di calcolo: gli approcci basati sulle meta-euristiche sono in grado di trovare lo stesso valore ottimale identificato con l'approccio a griglia in tempi significativamente inferiori; inoltre l'approccio GA è risultato maggiormente veloce rispetto a quello TS (in 3 dei 4 casi menzionati).
Sfortunatamente in un caso (dataset WBCP) nè l'approccio basato su GA nè quello su TS sono stati in grado di individuare il valore di fitness ottimale riscontrato dalla sperimentazione a griglia; le soluzioni fornite dalle meta-euristiche hanno richiesto tuttavia tempi di calcolo significativamente inferiori.
Infine, in corrispondenza dell'ultimo dataset (WBCD), la Model Selection basata su TS riesce ad individuare lo stesso valore di fitness ottimale rilevato dalla sperimentazione a griglia, ancora una volta con tempo di calcolo sensibilmente inferiore. Al contrario, la Model Selection basata su GA riesce ad individuare un livello di fitness lievemente

inferiore; i tempi di calcolo sono più bassi di quelli dell'approccio TS e ciò potrebbe far pensare ad una terminazione eccessivamente anticipata per l'approccio GA.

I risultati sono stati tuttavia incoraggianti e hanno giustificato l'utilizzo delle metaeuristiche di ricerca anche per le problematiche di Multiple Kernel Learning e di Combinazione di SVMs, rendendo così possibile l'implementazione del Framework di classificazione ad alto livello.

6.3.2 Framework basato su GA a caratteri discreti e su TS

In questa sezione sono riportati i migliori valori di fitness ottenuti dalla sperimentazione del Framework di classificazione ad alto livello proposto dal lavoro di ricerca (*Model Selection + Multiple Kernel Learning + Classifiers Combining*) e realizzato sia con GA a caratteri discreti che con TS. Per ridurre gli effetti della casualità, relativi all'inizializzazione delle 3 popolazioni per l'approccio GA e per la costruzione delle 3 soluzioni iniziali per TS, la sperimentazione è stata ripetuta 3 volte su ogni dataset.
Sono stati mantenuti i valori precedentemente previsti per la Model Selection: parametro di regolazione C, tipo di kernel, parametri γ e d per i kernel RBF e polinomiale. Sono stati inoltre previsti: $N = 3$ SVMs per il sistema di combinazione, $M = 3$ kernel basali per la SVM con Multiple Kernel Learning, e gli stessi 5 valori sia per i pesi del sistema di combinazione $\mathbf{w}$ che per i coefficienti della combinazione kernel a (0.1, 0.25, 0.5, 0.75, 1). Si è intenzionalmente deciso di non considerare il valore 0 poiché si desiderava espressamente avere combinazioni di 3 elementi (SVMs o kernel) senza dare la possibilità al Framework di effettuare delle selezioni.
Le configurazioni così ottenute sono:

- $(|C| \times |\mathbf{w}| \times (|\gamma| + |degree| + 1))^N = (4 \times 5 \times (25 + 4 + 1))^3 = 216'000'000$ per la combinazione di 3 SVMs;
- $|C| \times (|\mathbf{w}| \times (|\gamma| + |degree| + 1))^M = 4 \times (5 \times (25 + 4 + 1))^3 = 13'500'000$ per il Multiple Kernel Learning con 3 kernel basali;

per un totale di $120 + 216'000'000 + 13'500'000 = 229'500'120$ configurazioni per l'intero Framework.

Le dimensioni per ogni popolazione dell'approccio GA sono state fissate a: 10 individui per la Model Selection, 30 per il Multiple Kernel Learning e 60 per la Combinazione di SVMs. La probabilità di mutazione viene scelta casualmente nell'intervallo [0.001; 0.01], in modo indipendente per ogni popolazione. Infine i criteri di terminazione rimangono invariati rispetto a quanto definito nella precedente sezione e sono applicati ad ogni

popolazione.
Per quanto concerne l'approccio TS, i parametri relativi alla Model Selection rimangono identici a quelli specificati nella precedente sezione, mentre le lunghezze delle tabu-list sono fissate a 30 per il Multiple Kernel Learning e 60 per la combinazione di SVMs; il massimo numero di fasi di intensificazione senza miglioramento è fissato a 5 per entrambi i task, così come il massimo numero di fallimenti della diversificazione (20) ed il numero massimo di iterazioni complete (100).

Insieme ai risultati del Framework vengono riportati, come ulteriore parametro di confronto, i valori di fitness ottenuti dalla sperimentazione a griglia per la sola Model Selection. E' da sottolineare che le configurazioni totali del Framework sono talmente numerose da richiedere tempi di calcolo improponibili per un approccio a griglia. E' possibile ottenerne una stima considerando che il tempo impiegato per la Model Selection a griglia è quello richiesto per addestrare tutte le configurazioni SVM basali. Un'implementazione "brutale" della griglia, per il Framework, richiederebbe che tutte le configurazioni SVM basali vengano addestrate un egual numero di volte: è quindi possibile considerare il tempo medio di addestramento di una SVM, ottenuto a partire dai tempi dalla Model Selection a griglia, e moltiplicare tale tempo per:

- il numero di configurazioni e per N (numero di classificatori da combinare) nel caso della Combinazione di SVMs;
- il numero di configurazioni previste per il Multiple Kernel Learning. Va tuttavia considerato che in questo caso il tempo risultante è una stima per difetto, visto che il Multiple Kernel Learning prevede un *overhead* dovuto ai tempi di calcolo per combinare le N matrici kernel, per ogni SVM da addestrare.

In tabella 6.4 è riportata, per ogni dataset, la stima dei tempi necessari alla sperimentazione dell'intero Framework con approccio a griglia.

	Giorni	Ore	Minuti	Secondi
WEATHER	575	12	30	26
WEATHER (no norm.)	575	12	30	26
SPECTF	752	14	30	34
PARKINSONS	1195	7	30	54
WBCP	1416	16	1	4
WBCD	6087	5	49	35

Tabella 6.4: Stima dei tempi per un'implementazione a griglia del Framework

E' impressionante pensare che nel caso migliore un anno di sperimentazione a griglia non sarebbe sufficiente, mentre nel caso peggiore ne occorrerebbero addirittura oltre 16.

In tabella 6.5 sono riportati i migliori risultati ottenuti sulle 3 iterazioni della sperimentazione del Framework basato su GA a caratteri discreti e su TS, mentre in tabella 6.6 sono riportati i risultati medi, sia in termini di fitness che di tempi di calcolo.

	GA-based Framework		TS-based Framework		Grid-Model Selection
	Fitness	Tempo	Fitness	Tempo	Fitness
WEATHER	100.00%	***0:03:41***	100.00%	0:13:36	94.44%
WEATHER (no norm.)	64.44%	***0:10:14***	64.44%	0:30:19	50.00%
SPECTF	83.75%	***0:15:42***	83.75%	0:39:03	80.00%
PARKINSONS	***95.51%***	***0:43:28***	95.20%	1:10:59	93.09%
WBCP	74.02%	***1:14:49***	74.02%	1:41:52	70.42%
WBCD	98.07%	***1:42:10***	***98.11%***	3:32:33	97.55%

TABELLA 6.5: Dataset pubblici - Migliori fitness su 3 iterazioni, per Framework basato su GA a caratteri discreti e su TS

Dalla tabella si evince immediatamente il primo vantaggio offerto dalla soluzione proposta: i valori di fitness ottenuti con il classico approccio a griglia vengono superati in tutti i casi dal Framework di classificazione ad alto livello, sia esso basato su GA o TS. Ciò è naturalmente dovuto ai benefici propri dei sistemi di combinazione di classificatori e del Multiple Kernel Learning, in grado di migliorare le prestazioni rispetto ai singoli classificatori basali del sistema di combinazione (prima strategia) o ad una SVM con singolo kernel basale (seconda strategia).
Scendendo nei dettagli, si può osservare che il valore di fitness ottimale individuato dalle strategie GA e TS è lo stesso per ben 4 casi su 6, mentre sui 2 rimanenti GA individua un ottimo migliore rispetto a TS sul dataset PARKINSONS e, viceversa, TS individua un fitness migliore rispetto a GA sul dataset WBCD.
Un aspetto molto interessante sono i tempi di calcolo, sensibilmente inferiori per il Framework basato su GA rispetto a quello basato su TS. Un punto cruciale è relativo proprio al dataset WBCD, per il quale la strategia TS individua un fitness maggiore rispetto a GA di soli 0.04 punti percentuali (un'istanza positiva in meno e due istanze negative in più), ma in circa il doppio del tempo. Al contrario, nel caso del dataset PARKINSONS, la strategia GA individua un fitness maggiore di quello individuato da TS di 0.30 punti percentuali (un'istanza positiva in meno e 4 negative in più), e sempre con tempi di calcolo sensibilmente inferiori.

Come già accennato, per ovviare al fattore casualità, che può inficiare pesantemente sulla ricerca del fitness ottimale, le sperimentazioni sono state ripetute per 3 volte su ogni dataset; lo studio delle prestazioni medie può far comprendere meglio gli aspetti favorevoli legati all'implementazione basata su GA rispetto a quella basata su TS.
Osservando la tabella è facile notare come in realtà, mediamente, l'approccio basato su GA abbia avuto un comportamento più efficiente rispetto a TS: in realtà solo su 2 dataset le due strategie hanno individuato lo stesso valore di fitness ottimale (WEATHER con

	GA-based Framework		TS-based Framework		Tempi medi		t_{GA}/t_{TS}
	Fitness medio	dev.st	Fitness medio	dev.st	GA	TS	
WEATHER	100.00%	0.00%	100.00%	0.00%	0:04:55	0:17:53	0.28
WEATHER (no norm.)	***62.96%***	2.57%	61.11%	5.77%	0:14:12	0:31:12	0.45
SPECTF	***83.33%***	0.72%	82.92%	0.72%	0:18:11	0:36:23	0.50
PARKINSONS	***95.30%***	0.18%	94.25%	1.31%	0:40:27	1:09:21	0.58
WBCP	74.02%	0.00%	74.02%	0.00%	1:33:00	1:44:57	0.89
WBCD	98.04%	0.06%	***98.08%***	0.03%	2:11:01	3:27:58	0.63

Tabella 6.6: Dataset pubblici - Fitness e tempi medi su 3 iterazioni del Framework basato su GA a caratteri discreti e su TS

normalizzazione e WBCP), con valore costante su ognuna delle 3 diverse iterazioni (deviazione standard pari a 0%). GA ha individuato valori di fitness ottimale che risultano in media maggiori rispetto a quelli di TS, tranne per il caso del dataset WBCD, per il quale si nota ancora una differenza (in questo caso media) di 0.04 punti percentuali in favore di TS.
I tempi medi di calcolo rimangono tuttavia sensibilmente inferiori per GA: la colonna t_{GA}/t_{TS} nella tabella 6.6 indica il rapporto tra il tempo utilizzato da GA per individuare il fitness ottimale rispetto a quello impiegato da TS.

Infine, uno degli aspetti più importanti che giustifica l'effettiva validità della realizzazione di un Framework di classificazione ad alto livello si evince dalla visione dei macro-modelli di classificazione in corrispondenza dei quali sono stati ottenuti i valori di fitness ottimali. In tabella 6.7 sono sintetizzati i risultati in merito: non esiste una strategia "assoluta" per ottenere il modello decisionale più affidabile (Multiple Kernel Learning o Sistema di Combinazione di Classificatori), la scelta sembra essere dettata essenzialmente dal problema decisionale e dai dati a disposizione.

	GA-based Framework	TS-based Framework
WEATHER	Combinazione di SVMs	Combinazione di SVMs
WEATHER (no norm.)	SVM con Multiple Kernel Learning	SVM con Multiple Kernel Learning
SPECTF	SVM con Multiple Kernel Learning	SVM con Multiple Kernel Learning
PARKINSONS	Combinazione di SVMs	Combinazione di SVMs
WBCP	SVM con Multiple Kernel Learning	SVM con Multiple Kernel Learning
WBCD	SVM con Multiple Kernel Learning	SVM con Multiple Kernel Learning

Tabella 6.7: Dataset pubblici - Soluzioni migliori fornite dal Framework basato su GA a caratteri discreti e su TS

Dalle sperimentazioni è infatti scaturito che per i dataset WEATHER (senza normalizzazione) e PARKINSONS il macro-modello decisionale più efficace è un sistema di combinazione tra SVMs, mentre in tutti gli altri casi è una SVM con Multiple Kernel Learning.
La dipendenza tra dati e tipo di modello decisionale è ulteriormente confermato dai risultati ottenuti su WEATHER: a seconda se il dataset venga pre-processato o meno

attraverso normalizzazione, il macro-modello più affidabile risulta essere una SVM con combinazione di kernel o un sistema di combinazione tra SVMs a kernel basali.
E' essenziale notare come le due diverse implementazioni del Framework (con GA a caratteri discreti e TS) siano in accordo sulla scelta della tipologia di macro-modello per ogni problema decisionale, anche se è stato già osservato come, in corrispondenza di alcuni dataset, individuino valori differenti di fitness ottimale.

6.3.3 Framework II - GA con caratteri misti

Un ulteriore passo previsto dalla sperimentazione è l'utilizzo di una seconda definizione del Framework basato su GA, più precisamente si tratta dell'utilizzo di caratteri continui per ottimizzare:

- il parametro γ dei kernel RBF (nell'intervallo [0.001; 1000]),
- il parametro di regolazione C (nell'intervallo [1; 10]),
- i pesi per il sistema di combinazione di SVMs (nell'intervallo [0.1; 1]),
- i coefficienti di combinazione per il Multiple Kernel Learning (nell'intervallo [0.1; 1])

con la precisione fissata a 4 cifre decimali.
Il numero di possibili soluzioni di conseguenza "esplode" rispetto alla formulazione a soli caratteri discreti; a tal proposito si è ritenuto vantaggioso testare differenti valori per la probabilità di mutazione ed interagire con il criterio di terminazione legato al numero di generazioni consecutive senza miglioramento del fitness, con il fine di aumentare il peso del meccanismo di exploration delle soluzioni.
Essendo il numero di soluzioni molto elevato si è inizialmente deciso di fissare a 0.01 la probabilità di mutazione, assegnando al meccanismo di exploiration il livello massimo attribuibile (in maniera casuale) nella precedente sperimentazione basata su GA a soli caratteri discreti. Per il criterio di terminazione, basato sul mancato miglioramento del fitness, il numero di generazioni consecutive viene aumentato a 30.
In un secondo momento si è deciso di incrementare ulteriormente il meccanismo di exploiration, portando a 0.2 la probabilità di mutazione, e testando due differenti livelli per la terminazione: 20 e 30 generazioni consecutive senza miglioramento del fitness.

In tabella 6.8 vengono messi a confronto i migliori valori di fitness ottenuti dalla sperimentazione del Framework basato su GA a caratteri discreti, TS ed infine GA a caratteri misti.

	GA-based Framework		TS-based Framework		GA-based Framework II	
	Fitness	**Tempo**	**Fitness**	**Tempo**	**Fitness**	**Tempo**
WEATHER	100.00%	0:03:41	100.00%	0:13:36	100.00%	***0:01:50***
WEATHER (no norm.)	64.44%	***0:10:14***	64.44%	0:30:19	64.44%	0:20:32
SPECTF	83.75%	***0:15:42***	83.75%	0:39:03	83.75%	0:47:32
PARKINSONS	95.51%	0:43:28	95.20%	1:10:59	***96.22%***	2:08:40
WBCP	***74.02%***	***1:14:49***	***74.02%***	1:41:52	73.21%	3:40:04
WBCD	98.07%	1:42:10	***98.11%***	3:32:33	97.93%	3:24:19

Tabella 6.8: Dataset pubblici - Migliori fitness su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

Per i primi 3 dataset (WEATHER, con e senza normalizzazione, e SPECTF) i tre approcci individuano lo stesso valore di fitness ottimale. I tempi sono naturalmente minori per l'approccio GA a caratteri discreti che, oltre ad essere risultato più veloce dell'approccio basato su TS, deve ricercare il fitness ottimale all'interno di uno spazio di ricerca delle soluzioni molto più ristretto rispetto all'implementazione basata su caratteri misti; solo nel primo caso (WEATHER normalizzato) la nuova implementazione riesce a trovare il valore ottimale in tempo minore, ma ciò potrebbe essere semplicemente dovuto a fattori casuali (inizializzazione delle popolazioni).
In corrispondenza del dataset PARKINSONS, sul quale l'implementazione GA a caratteri discreti era già stata in grado di individuare un valore di fitness maggiore rispetto a TS, la nuova implementazione basata su GA a caratteri misti offre un ulteriore margine di miglioramento, come in linea teorica ci si aspettava dall'approccio proposto. L'aumento del tempo di calcolo è un "contro" più che atteso: anche se la nuova definizione del Framework impiega circa il doppio del tempo rispetto a quello più alto finora registrato (in corrispondenza della TS), ciò può essere trascurato, sia in relazione al fatto di aver ottenuto un miglioramento in termini di prestazioni (0.50 punti percentuali rispetto al migliore GA e ben 1 punto rispetto a TS) e sia rispetto alla considerazione che lo spazio di ricerca per l'implementazione a caratteri continui è estremamente più ampio.
Gli ultimi due dataset (WBCP e WBCD) non confermano tuttavia le buone aspettative finora evidenziate per la nuova implementazione proposta per il Framework. TS e GA a caratteri discreti riescono infatti ad individuare lo stesso valore di fitness ottimale per il datset WBCP (GA lo fa in tempi molto ristretti), mentre l'approccio basato su GA a caratteri misti individua un valore inferiore e con un dispendio di tempo di calcolo sensibilmente maggiore.
La situazione è ancora peggiore in corrispondenza del dataset WBCD: nonostante il nuovo approccio impieghi circa lo stesso tempo dell'implementazione basata su TS (in corrispondenza della quale si è ottenuto il miglior livello di fitness) l'ottimo individuato è addirittura inferiore a quello ottenuto dalla sperimentazione dell'approccio GA a caratteri discreti.

Un confronto maggiormente significativo si ottiene incrociando i risultati riportati nelle due tabelle 6.9 e 6.10, relativi rispettivamente ai valori medi di fitness e tempi di calcolo registrati su 3 iterazioni per ogni dataset.

	GA-based Framework		TS-based Framework		GA-based Framework II	
	Fitness medio	dev.st	Fitness medio	dev.st	Fitness medio	dev.st
WEATHER	100.00%	0.00%	100.00%	0.00%	100.00%	0.00%
WEATHER (no norm.)	62.96%	2.57%	61.11%	5.77%	***64.44%***	0.00%
SPECTF	***83.33%***	0.72%	82.92%	0.72%	82.50%	1.25%
PARKINSONS	95.30%	0.18%	94.25%	1.31%	***95.75%***	0.53%
WBCP	***74.02%***	0.00%	***74.02%***	0.00%	72.39%	1.14%
WBCD	98.04%	0.06%	***98.08%***	0.03%	97.90%	0.06%

TABELLA 6.9: Dataset pubblici - Fitness medi su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

	GA-based Framework	TS-based Framework	GA-based Framework II
WEATHER	0:04:55	0:17:53	0:02:10
WEATHER (no norm.)	0:14:12	0:31:12	0:30:57*
SPECTF	0:18:11*	0:36:23	0:40:04
PARKINSONS	0:40:27	1:09:21	1:58:19*
WBCP	1:33:00*	1:44:57*	2:17:50
WBCD	2:11:01	3:27:58*	4:06:54

TABELLA 6.10: Dataset pubblici - Tempi medi su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

Osservando i valori medi le considerazioni fatte sui migliori risultati vanno leggermente riviste. Viene confermato il dato che la nuova implementazione fornisce le stesse prestazioni delle precedenti sul dataset WEATHER normalizzato, mantenendo tra l'altro il più basso tempo di calcolo; sullo stesso dataset, ma senza normalizzazione, offre le prestazioni migliori: individua su tutte le 3 iterazioni lo stesso valore di fitness ottimale con tempo medio di calcolo praticamente identico a quello impiegato dall'implementazione basata su TS.

In corrispondenza del dataset SPECTF non è in grado di offrire prestazioni simili agli altri due approcci ed impiega il tempo di calcolo più elevato, anche se dell'ordine di quello di TS.

Sul dataset PARKINSONS conferma le buone prestazioni riscontrate per i risultati migliori: il fitness ottimale individuato è (mediamente) il più alto tra gli approcci ed il tempo di calcolo è meno del doppio di quello per TS, rimanendo comunque accettabile (circa 2 ore).

Per gli ultimi due dataset (WBCP e WBCD) valgono le medesime considerazioni fatte in relazione ai migliori risultati.

6.4 Risultati - Dataset da attività di ricerca

6.4.1 Framework basato su GA a caratteri discreti e su TS

In questa sezione sono riportati i risultati relativi alla sperimentazione del Framework di classificazione ad alto livello realizzato con gli approcci GA a soli caratteri discreti e TS, in riferimento ai dataset privati. Le condizioni sperimentali sono identiche a quelle riportate nell'omonima sezione relativa alla sperimentazione sui dataset pubblici (sezione 6.3.2). In tabella 6.11 sono sintetizzati i migliori risultati ed inoltre i valori di fitness ottenuti dalla sperimentazione a griglia per la sola problematica di Model Selection.

	GA-based Framework		**TS-based Framework**		**Grid-Model Selection**
	Fitness	**Tempo**	**Fitness**	**Tempo**	**Fitness**
Noduli Tiroidei	80.15%	***0:15:00***	***82.75%***	0:42:11	78.99%
Musica ed Emozioni	***59.94%***	***0:45:46***	57.98%	0:55:29	52.22%
EDDC	***87.35%***	***1:42:40***	85.56%	2:51:44	78.84%

TABELLA 6.11: Dataset privati - Migliori fitness su 3 iterazioni, per Framework basato su GA a caratteri discreti e su TS

E' possibile notare che i risultati ottenuti sono in linea con quelli preliminari riscontrati sui dataset pubblici: il Framework ha naturalmente prestazioni superiori a quelle ottenibili dai singoli classificatori basali e rilevate attraverso la sperimentazione a griglia. L'approccio GA a soli caratteri discreti è risultato sensibilmente più veloce di TS in tutti i casi, ma maggiormente efficace su 2 dei 3 dataset.

Anche per questa sessione sperimentale un confronto più dettagliato tra le implementazioni può essere effettuato considerando le prestazioni medie su 3 iterazioni per ogni dataset, sia in termini di fitness che di tempi di calcolo. Nella seguente tabella 6.12 sono riportati i relativi risultati. Le considerazioni che è possibile trarre dai risultati medi

	GA-based Framework		**TS-based Framework**		**Tempi medi**		$\mathbf{t_{GA}/t_{TS}}$
	Fitness medio	**dev.st**	**Fitness medio**	**dev.st**	**GA**	**TS**	
Noduli Tiroidei	80.15%	0.00%	***81.19%***	1.37%	0:20:51	0:43:18	0.48
Musica ed Emozioni	***59.01%***	1.62%	56.73%	2.17%	0:48:17	0:52:40	0.92
EDDC	***85.66%***	2.49%	84.41%	1.05%	1:54:37	2:40:27	0.71

TABELLA 6.12: Dataset privati - Fitness e tempi medi su 3 iterazioni del Framework basato su GA a caratteri discreti e TS

sono praticamente identiche a quelle relative alle migliori prestazioni: l'approccio TS risulta più efficace nel solo caso del dataset NODULI TIROIDEI, sebbene con tempi di calcolo sensibilmente superiori a GA. Sugli altri due dataset (MUSICA&EMOZIONI ed EDDC) l'implementazione del Framework basata su GA a soli caratteri discreti risulta non solo più efficace ma anche più efficiente, con tempi di calcolo inferiori rispetto a TS.

Infine è anche utile osservare quali siano le tipologie di macro-modello indicate come ottimali dalle due differenti implementazioni del Framework (tabella 6.13). GA indica la combinazione di SVMs come soluzione ottimale per tutti i dataset, mentre TS risulta in disaccordo in corrispondenza del solo dataset EDDC, per il quale indica una SVM con Multiple Kernel Learning come macro-modello ottimale.

Va sottolineato che la sperimentazione sui 3 dataset reali conferma quanto scaturito dalla preliminare analisi sui dataset pubblici: la scelta tra le strategie di combinazione di classificatori o di Multiple Kernel Learning è strettamente dettata dal problema decisionale e dai dati a disposizione. Infatti, mentre il Multiple Kernel Learning è risultata la strategia più frequentemente indicata come ottimale nella sperimentazione sui dataset pubblici, ciò viene ribaltato in corrispondenza dei dataset reali, per i quali viene "preferita" la strategia di combinazione tra SVMs.

	GA-based Framework	**TS-based Framework**
Noduli Tiroidei	Combinazione di SVMs	Combinazione di SVMs
Musica ed Emozioni	Combinazione di SVMs	Combinazione di SVMs
EDDC	Combinazione di SVMs	SVM con Multiple Kernel Learning

TABELLA 6.13: Soluzioni migliori fornite dal Framework basato su GA a caratteri discreti e su TS

6.4.2 Framework II - GA con caratteri misti

La presente sperimentazione è relativa all'utilizzo del Framework implementato con GA a caratteri misti. Tutti i vari parametri caratterizzanti la sperimentazione sono identici a quelli riportati nell'omonima sezione relativa ai dataset pubblici (sezione 6.3.3).
In tabella 6.14 sono riportati i migliori risultati relativi al fitness e al tempo di calcolo per le varie implementazioni del Framework (TS, GA a caratteri discreti e misti).

	GA-based Framework		**TS-based Framework**		**GA-based Framework II**	
	Fitness	**Tempo**	**Fitness**	**Tempo**	**Fitness**	**Tempo**
Noduli Tiroidei	80.15%	0:15:00	82.75%	0:42:11	***84.18%***	1:52:24
Emozioni e Musica	***59.94%***	***0:45:46***	57.98%	0:55:29	***59.94%***	1:59:23
EDDC	***87.35%***	1:42:40	85.56%	2:51:44	86.49%	3:02:39

TABELLA 6.14: Dataset privati - Migliori fitness su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

Sui dataset reali sono stati ottenuti i risultati (migliori) più interessanti per il lavoro di ricerca. Il Framework basato su GA a caratteri misti individua infatti valori di fitness ottimali maggiori rispetto all'approccio TS, su tutti i dataset, pur richiedendo un maggior tempo di calcolo. In tal modo si riesce ad ottemperare all'inefficacia riscontrata

su NODULI TIROIDEI da parte del Framework implementato con GA a soli caratteri discreti: si guadagna circa 1.5 punti percentuali di fitness rispetto a TS e ben 4 punti percentuali rispetto a GA a soli caratteri discreti.
In corrispondenza del dataset MUSICA&EMOZIONI entrambe le implementazioni GA individuano lo stesso valore ottimale di fitness (maggiore rispetto a TS), in tempi sensibilmente inferiori per GA a soli caratteri discreti.
Nel caso del dataset EDDC il valore di fitness ottimale più alto è stato riscontrato in corrispondenza dell'implementazione con GA a soli caratteri discreti ed in tempo sensibilmente inferiore rispetto alle altre 2 implementazioni. Il valore individuato da GA a caratteri misti è comunque superiore a quello ottenuto da TS, e con un onere temporale comunque trascurabile a confronto (10 minuti in più su circa 3 ore di elaborazione).

Nelle due seguenti tabelle 6.15 e 6.16 sono riportati i valori medi di fitness e tempo di calcolo su 3 iterazioni per ogni dataset. Va ricordato che, per quanto concerne l'implementazione del Framework con GA a caratteri misti, le 3 iterazioni non sono identiche poiché sono stati adottati differenti valori di probabilità di mutazione e dimensione delle popolazioni (come specificato nell'omonima sezione 6.3.3 relativa ai dataset pubblici).

	GA-based Framework		**TS-based Framework**		**GA-based Framework II**	
	Fitness medio	**dev.st**	**Fitness medio**	**dev.st**	**Fitness medio**	**dev.st**
Noduli Tiroidei	80.15%	0.00%	81.19%	1.37%	***83.04%***	0.98%
Musica ed Emozioni	***59.01%***	1.62%	56.73%	2.17%	58.20%	1.52%
EDDC	***85.66%***	2.49%	84.41%	1.05%	83.92%	2.33%

Tabella 6.15: Dataset privati - Fitness medi su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

	GA-based Framework	**TS-based Framework**	**GA-based Framework II**
Noduli Tiroidei	0:20:51	0:43:18	1:22:53*
Emozioni e Musica	0:48:17*	0:52:40	1:18:54
EDDC	1:54:37*	2:40:27	2:42:07

Tabella 6.16: Dataset privati - Tempi medi su 3 iterazioni: confronto tra Framework basato su GA a caratteri discreti, TS e GA a caratteri misti

Le considerazioni tratte per le migliori prestazioni sono valide anche per quelle medie: le implementazioni basate su GA (a caratteri discreti e misti complessivamente), sono risultate più efficaci di quella basata su TS (in particolare GA a caratteri misti in corrispondenza del dataset NODULI TIROIDEI).
L'unica differenza rispetto ai migliori risultati è che, nel caso dei valori medi, l'implementazione TS risulta più efficace di GA a caratteri misti sul dataset EDDC; tuttavia GA a soli caratteri discreti fornisce il miglior valore di fitness medio.
In merito ai tempi di calcolo si può notare che:

- GA a caratteri discreti è più veloce rispetto alle altre due implementazioni;
- TS e GA a caratteri misti hanno tempi di calcolo comparabili. La differenza maggiore si ha in corrispondenza del dataset NODULI TIROIDEI, ma è un contro trascurabile considerato il fatto che in corrispondenza di tale dataset il più alto valore medio di fitness è fornito proprio dall'implementazione di GA a caratteri misti.

6.5 Kernel Learning con ACO, GA e TS

In questa ultima sezione sono riportati i risultati in merito ad un'ulteriore tema di ricerca strettamente collegato al Framework di classificazione ad alto livello proposto: l'implementazione del task di Multiple Kernel Learning attraverso ACO. I risultati sono inoltre confrontati con quelli ottenuti attraverso le altre meta-euristiche, che sono state utilizzate per implementare il Framework di classificazione ad alto livello. L'obiettivo è valutare la validità della nuova tecnica rispetto agli altri approcci.
In tabella 6.17 vengono sintetizzati i risultati.

	GA-based MKL	**GA-II-based MKL**	**TS-based MKL**	**ACO-based MKL**
Dataset pubblici				
WEATHER	***100.00%***	94.44%	***100.00%***	***100.00%***
WEATHER (no norm.)	***64.44%***	***64.44%***	***64.44%***	58.89%
SPECTF	83.75%	83.75%	83.75%	83.75%
PARKINSONS	94.11%	94.47%	***94.81%***	***94.81%***
WBCP	***74.02%***	72.88%	***74.02%***	73.69%
WBCD	98.07%	97.83%	***98.11%***	97.97%
Dataset da attività di ricerca				
Noduli Tiroidei	***79.26%***	***79.26%***	***79.26%***	78.99%
Musica ed Emozioni	***58.34%***	57.29%	57.23%	52.22%
EDDC	***85.56%***	83.32%	***85.56%***	80.05%

Tabella 6.17: Multiple Kernel Learning - Fitness ottimale calcolato con GA a caratteri discreti e misti, TS e ACO

Dalla tabella si evince che la tecnica ACO non sembra essere in grado di offrire gli stessi vantaggi delle altre meta-euristiche: al più in 3 casi, relativi ai dataset pubblici, fornisce lo stesso valore di fitness ottimale individuato da almeno una delle altre meta-euristiche (nello specifico: WEATHER, SPECTF e PARKINSONS). In altri casi riesce a fornire un valore di fitness ottimale lievemente inferiore a quello individuato dalle altre tecniche e comunque comparabile agli altri (WBCP, WBCD e NODULI TIROIDEI). Infine, per

i rimanenti 3 casi (WEATHER non normalizzato, MUSICA&EMOZIONI e EDDC) i valori indicati da ACO sono purtroppo sensibilmente inferiori a quelli indicati da tutti gli altri approcci.

Capitolo 7

Conclusioni

Il lavoro di ricerca ha proposto l'utilizzo degli algoritmi genetici (opportunamente modificati rispetto alla loro formulazione classica) per effettuare una ricerca efficiente delle soluzioni ottimali di tre differenti problematiche di classificazione a Support Vector Machines: Model Selection (individuazione della migliore configurazione dei parametri di un singolo classificatore), Multiple Kernel Learning (costruzione del kernel più appropriato come combinazione di kernel basali, per un singolo classificatore) e SVMs Ensemble (sistema di combinazione tra differenti classificatori Support Vector Machines).
Le tre problematiche sono attualmente trattate in modo completamente separato ed indipendente dalla comunità scientifica di riferimento, ed afferiscono a due diverse scuole di pensiero: migliorare le capacità di una specifica tecnica di addestramento, ad esempio SVM (Model Selection e Multiple Kernel Learning), oppure sfruttare le differenze di più classificatori al fine di costruire un sistema di combinazione con un'elevata affidabilità globale (SVMs Ensebmle). L'innovazione proposta nel presente lavoro consiste nell'unificare i due approcci sotto un unico Framework, visto che è difficile, se non impossibile, stabilire a priori quali delle due strategie sia in grado di individuare il modello di classificazione più affidabile in riferimento ad uno specifico problema.
La ricerca della migliore soluzione del Framework è caratterizzata da un elevatissimo numero di configurazioni da testare, rendendo il classico approccio "a griglia" inefficiente se non addirittura impraticabile; ciò motiva l'utilizzo di un'opportuna meta-euristica di ricerca. Lo studio si è concentrato particolarmente sull'uso degli algoritmi genetici, proponendo alcune modifiche specifiche per il setting realizzato, ma si è tenuto conto anche di altre tecniche di ricerca che hanno già trovato applicazione soprattutto nel dominio della Model Selection per SVMs, ovvero le meta-euristiche Tabu-Search e Ant Colony Optimization. In riferimento a quest'ultima è stato inoltre proposto un approccio per la realizzazione del task di Multiple Kernel Learning, approccio al momento assolutamente innovativo rispetto al relativo stato dell'arte (i lavori attuali sono principalmente rivolti

alle attività di Feature Selection o di Model Selection).
Le variazioni proposte all'approccio algoritmi genetici sono dettagliatamente descritte nel capitolo 4, basti ricordare che il lavoro propone di abbandonare la rappresentazione in bit per i singoli geni ottenendo un migliore controllo sulle configurazioni SVM che si vogliono testare. Questo primo accorgimento permette infatti di ottenere una stretta analogia con la classica sperimentazione "a griglia", mentre con la rappresentazione in bit sorgerebbero difficoltà quali l'utilizzo di funzioni di trasformazione che potrebbero non rispettare le configurazioni da testare o problemi tra il numero di valori rappresentabili da un insieme di geni ed i valori realmente attribuibili alla componente del fenootipo relativa a quell'insieme di geni (cfr. sezione 4.1).
Mentre la prima variazione esposta utilizza cromosomi con sole componenti discrete, una seconda variazione prevede l'uso di cromosomi che abbiano componenti a natura mista, nominali e continue, rispecchiando in modo più corretto le possibili configurazioni di un classificatore SVM: i tipi di kernel o il grado del kernel polinomiale sono "caratteri" discreti (strettamente nominale ed intero, rispettivamente), mentre il parametro di regolazione C o il parametro γ del kernel RBF possono essere definiti come "caratteri" continui o anche discreti nel caso in cui si vogliano testare solo alcuni specifici valori.

I risultati confermano che l'uso dell'approccio algoritmi genetici risulta efficace e particolarmente efficiente rispetto alla meta-euristica Tabu-Search. Mediamente, già la sola implementazione algoritmi genetici a "caratteri" discreti fornisce valori di ottimo (accuratezza bilanciata) maggiori rispetto a Tabu-Search, in particolare su ben 5 sperimentazioni delle 9 totali (3 su 6 relative ai dataset pubblici e 2 su 3 ai privati). Su 2 casi (dataset pubblici) le due metodologie sono in grado di trovare lo stesso valore ottimale di fitness, ma gli algoritmi genetici ci riescono in tempi decisamente inferiori (cfr. tabelle 6.6 e 6.12). In pratica solo in 2 casi (1 relativo ai dataset pubblici e 1 ai privati) Tabu-Search riesce ad individuare un ottimo migliore rispetto agli algoritmi genetici.
La definizione a "caratteri" misti per gli algortimi genetici non ha sfortunatamente fornito i promettenti risultati attesi, in alcuni casi l'ottimo individuato ha infatti un valore minore rispetto a quello scaturito dalla definizione a "caratteri" discreti o dalla Tabu-Serach (cfr. tabelle 6.9 e 6.15): sembra che, in alcuni casi, il meccanismo di *exploration* (mutazione) non sia sufficiente ad evitare che la tecnica converga verso un ottimo locale, nonostante siano state impostate probabilità di mutazione anche relativamente alte o sia stato aumentato il numero di individui all'interno della popolazione.
Va tuttavia sottolineato che, in altri casi, il valore ottimo che era stato individuato dalla definizione a "caratteri" discreti è stato superato da quella a "caratteri" misti e che, di conseguenza, l'eventuale combinazione di entrambe le definizioni garantisce una maggiore efficacia rispetto alla meta-euristica di ricerca Tabu-Search, incrementando di una unità il numero di successi del confronto (dataset NODULI TIROIDEI, cfr. tabella 6.15) ma

soprattutto incrementando i valori di fitness ottimali già individuati.
Un ultimo importante risultato è costituito dai relativi modelli ottimali: è infatti stato realmente osservato come non esista una scelta valida in senso generale o con maggiore frequenza, il macro-modello (SVM ottenuta con Multiple Kernel Learning o un sistema di SVMs) è dettato dal problema decisionale e/o dai dati a disposizione.

Infine, in merito alla tecnica Ant Colony Optimization per il Multiple Kernel Learning va sottolineato che attualmente essa presenta prestazioni scadenti rispetto alle altre meta-euristiche di ricerca (cfr. tabella 6.17), fornisce comunque un possibile spunto per ulteriori ricerche e, naturalmente, rappresenta una possibile applicazione non ancora sfruttata in merito alla specifica problematica.

Capitolo 8

Sviluppi Futuri

Sono differenti gli sviluppi futuri che il lavoro offre, sia rivolti ad obiettivi prettamente di ricerca scientifica che alla realizzazione di strumenti di supporto all'implementazione di task di classificazione.

Rimanendo nell'ambito della metodologia Support Vector Machines sarebbe utile sfruttare anche i più recenti lavori su algoritmi di apprendimento efficienti, visto che la parte più onerosa e *time consuming* riguarda appunto l'addestramento dei diversi classificatori del Framework.
Sarebbe inoltre possibile prevedere nuove sessioni sperimentali del Framework impostando ulteriori parametri da ottimizzare, in particolare si potrebbe incorporare il task di Feature Selection (strategia *wrapper*) o effettuare la selezione degli operatori di combinazione dei kernel per il task di Multiple Kernel Learning o, ancora, ottimizzare la strategia del sistema di combinazione dei classificatori (il lavoro è stato basato unicamente su un sistema di voto pesato, con ottimizzazione dei pesi).

Da un punto di vista prettamente implementativo, gli sviluppi futuri riguardano il mantenimento e l'aggiornamento della libreria java per gli algoritmi genetici. Tale libreria può costituire un utile strumento di supporto per analisi di dati e sarà sicuramente utilizzata nelle attività del Laboratorio di Ingegneria delle Decisioni per i Servizi Sanitari, non solo come strumento di analisi (nella versione realizzata e presentata nella sezione 4.3) ma anche per ulteriori sviluppi e scopi di ricerca.
Infine, l'architettura del sistema realizzato, che pone WEKA come strumento per l'addestramento dei classificatori SVM, verrà ulteriormente ampliata in modo da poter utilizzare non soltanto la metodologia Support Vector Machines ma anche tutte le altre tecniche di classificazione messe a disposizione da WEKA. Tale nuovo sistema fornirebbe ulteriori potenzialità, andando a sfruttare le diverse capacità di rappresentazione della

conoscenza proprie di ogni metodologia, aumentando la probabilità di ottenere un modello decisionale affidabile dai task di Model Selection e di combinazione di classificatori: nel primo caso il tipo di tecnica di apprendimento diviene un parametro da ottimizzare, mentre nel secondo si ha la possibilità di combinare tra loro schemi molto differenti di rappresentazione della conoscenza.

Bibliografia

[1] T.G. Dietterich. Ensemble methods in machine learning. *in J. Kittler and F. Roli editors. Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy, Spinger-Verlag*, pages 1–15, 2000.

[2] N.C. de Condorcet. Essai sur l'application de l'analyse a la probabilité des decisions rendues a la pluralité des voix. *Imprimerie Royale*, 1785.

[3] G. Valentini and F. Masulli. Ensembles methods. *In Neural Nets, Springer Berlin/Heidelberg*, 2486 of Lecture Notes in Computer Science:3–20, 2002.

[4] F. Kimura and M. Shridar. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, 24(10):969–983.

[5] L. Lam and C. Sue. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man and Cybernetics*, 27(5):553–568.

[6] M.P. Perrone and L.N. Cooper. When networks disagree: ensemble methods for hybrid neural networks. *In R.J. Mammone editor, Artificial Neural Networks for Speech and Vision*, pages 126–142, 1993.

[7] L. Lam and C. Sue. Optimal combination of pattern classifiers. *Pattern Recognition Letters*, pages 945–954, 1995.

[8] M. Breukelen van, R.P.W. Duin, D. Tax, and J.E. Hartog den. Combining classifiers for the recognition of handwritten digits. *In Ist IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition*, pages 13–18, 1997.

[9] L.I. Kuncheva. An application of owa operators to the aggragation of multiple classification decisions. *In The Ordered Weighted Averaging operators. Theory and Applciations*, pages 330–343, 1997.

[10] R.E. Schapire. The strenght of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[11] B. Efron and R. Tibshirani. An introduction to the bootstrap. *Chapman and Hall, New York*, 1993.

[12] L. Breiman. Bagging predictors. *Pattern Recognition*, 24(2):123–140, 1996.

[13] M. Skurichina and R.P.W. Duin. Bagging for linear classifiers. *Pattern Recognition*, 31(7):909–930, 1998.

[14] M. Skurichina and R.P.W. Duin. Bagging and the random subspace method for redundant feature spaces. *In Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, 2096 of Lecture Notes in Computer Science:1–10, Springer-Verlag, 2001.

[15] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. *In Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.

[16] R.E. Schapire. A brief introduction to boosting. *In Thomas Dean, editor, 16th International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999.

[17] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1): 119–139, 1997.

[18] H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks. *In Advances in Neural Information Processing Systems*, 10:647–653, 1998.

[19] R. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[20] H. Drucker and C. Cortes. Boosting decision trees. *In Advances in Neural Information Processing Systems*, 8, 1996.

[21] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[22] H. Drucker, C. Cortes, L. Jackel, Y. LeCun, , and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[23] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[24] L. Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7): 1493–1517, 1999.

[25] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[26] J. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 39(5), 2001.

[27] B. Parmanto, P. Munro, and H. Doyle. Improving committe diagnosis with resampling techniques. *In D.S. Touretzky, M. Mozer, and M. Hesselmo, editors, Advances in Neural Information Processing Systems*, 8:882–888, 1996.

[28] B. Parmanto, P. Munro, and H. Doyle. Reducing variance of committee predition with resampling techniques. *Connection Science*, 8(3/4):405–416, 1996.

[29] A. Sharkey, N. Sharkey, and G. Chandroth. Diverse neural net solutions to a fault diagnosis problem. *Neural Computing and Applications*, 4:218–227, 1996.

[30] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[31] L.I. Kuncheva, F. Roli, G.L. Marcialis, and C.A. Shipp. Complexity of data subsets generated by the random subspace method: An experimental investigation. *In J. Kittler and F. Roli, editors, Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, 2096 of Lecture Notes in Computer Science: 349–358, 2001.

[32] M. Skurichina and R.P.W. Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications*, 5(2):121–135, 2002.

[33] L. Xu, C. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwritting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[34] H.S. Park and S.W. Lee. Off-line recognition of large sets handwritten characters with multiple hidden-markov models. *Pattern Recognition*, 29(2):231–244, 1996.

[35] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):125–130, 1991.

[36] R.A. Jacobs. Methods for combining experts probability assessment. *Neural Computation*, 7:867–888, 1995.

[37] M. Jordan and R. Jacobs. Hierarchies of adaptive experts. *In Advances in Neural Information Processing Systems*, 4:985–992, 1992.

[38] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[39] E. Mayoraz and M. Moreira. On the decomposition of polychotomies into dichotomies. *In The XIV International Conference on Machine Learning*, pages 219–226, 1997.

[40] F. Masulli and G. Valentini. Comparing decomposition methods for classification. *In R.J. Howlett and L.C. Jain, editors, KES'2000, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pages 788–791, 2000.

[41] R. Anand, G. Mehrotra, C.K. Mohan, and S. Ranka. Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, 6:117–124, 1995.

[42] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

[43] M. Moreira and E. Mayoraz. Improved pairwise coupling classifiers with correcting classifiers. *In C. Nedellec and C. Rouveirol, editors, Lecture Notes in Artificial Intelligence*, 1398:160–171, 1998.

[44] D.W. Opitz and J.W. Shavlik. Actively searching for an effective neural network ensemble. *Connection Science*, 8(3/4):337–353, 1996.

[45] F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. *In J. Kittler and F. Roli, editors, Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, 2096 of Lecture Notes in Computer Science:78–87, 2001.

[46] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[47] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[48] G. Giacinto and F. Roli. Dynamic classifier fusion. *In J. Kittler and F. Roli, editors, Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, 1857 of Lecture Notes in Computer Science:177–189, 2000.

[49] J.R. Quinlan. C4.5 programs for machine learning. *Morgan Kauffman*, 1993.

[50] Y. Raviv and N. Intrator. Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8(3/4):355–372, 1996.

[51] B. Scholkopf and A.J. Smola. Learning with kernels. support vector machines, regularization, optimization, and beyond (book). Massachusetts Institute of Tchnology, 2002.

[52] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46:131–159, 2002.

[53] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[54] N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines. *Cambridge,MA:Cambridge University Press*, 2000.

[55] V. Vapnik. The nature of statistical learning theory. *Berlin:Springer*, 1995.

[56] T.S. Jaakkola and D. Haussler. Probabilistic kernel regression models. *In Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[57] G. Wahba, Y. Lin, and H. Zhang. Generalized approximate crossvalidation for support vector machines: Another way to look at marginlike quantities. *In A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans (Eds.) - Advances in large margin classifiers - Cambridge,MA: MIT Press*, pages 297–309, 2000.

[58] M. Opper and O. Winther. Gaussian processes and svm: Mean field and leave-one-out. *In A.J. Smola, P.L. Bartlett, B. Scholkopf, D. Schuurmans (Eds.) - Advances in large margin classifiers - Cambridge, MA: MIT Press*, pages 311–326, 2000.

[59] V. Vapnik. Statistical learning theory. *New York:John Wiley & Sons*, 1998.

[60] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9), 2000.

[61] O. Chapelle and V. Vapnik. Model selection for support vector machines. *In Advances in neural information processing systems*, 1999.

[62] T. Joachims. Estimating the generalization performance of a svm efficiently. *In Proceedings of the International Conference on Machine Learning. San Mateo, CA:Morgan Kaufman*, 2000.

[63] J. Platt. Probabilities for support vector machines. *In A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans (Eds.) - Advances in large margin classifiers - Cambridge, MA: MIT Press*, 2000.

[64] H. Lutkephol. Handbook of matrices. *New York: Wiley & Sons*, 1996.

[65] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On optimizing kernel alignment. *Journal of Machine Learning Research*, 2002.

[66] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[67] O. Bousquet and D.J.L. Herrmann. On the complexity of learning the kernel matrix. *In Advances in Neural Information Systems, the MIT Press, Cambridge, MA, USA*, 15:415–422, 2003.

[68] D. Conforti and R. Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers & Operations Research*, doi:10.1016/j.cor.2009.02.018, 2009.

[69] G.E.P. Box and N.R. Draper. Evolutionary operation: Statistical method for process improvement. *John Wiley & Sons*, 1969.

[70] H.J. Bremermann. Optimization through evolution and recombination. *In Self-Organizing Systems*, 1962.

[71] J.L. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through simulated evolution. *John Wiley & Sons, New York*, 1966.

[72] J.H. Holland. Adaption in natural and artificial systems. *University of Michigan Press, Ann Arbor*, 1975.

[73] I. Rechenberg. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Frommann-Holzboog, Stoccarda*, 1973.

[74] C.L. Huang and C.J. Wang. A ga-based feature selection and parameters optimization for support vector machines. *Expert Systems with Applications*, 31:231–240, 2006.

[75] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):164–171, 2000.

[76] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2):44–49, 1998.

[77] S. Salcedo-Sanz, M. Prado-Cumplido, F. Perez-Cruz, and C Bousono-Calzon. Feature selection via genetic optimization. *Proceedings of the ICANN international conference on artificial neural networks, Madrid*, pages 547–552, 2002.

[78] H. Frohlich and O. Chapelle. Feature selection for support vector machines by means of genetic algorithms. *Proceedings of the 15th IEEE international conference on tools with artificial intelligence, Sacramento, CA, USA*, pages 142–148, 2003.

[79] S. Lessmann, R. Stahlbock, and S.F. Crone. Genetic algorithms for support vector machine model selection. *International Joint Conference on Neural Networks*, 2006.

[80] L.M. He, X.B. Yang, and F.S. Kong. Support vector machines ensemble with optimizing weights by genetic algorithm. *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*, 2006.

[81] W.W. Yan, Z.G. Chen, and H.H. Shao. Multi support vector machines decision mode and its application. *Journal of Shanghai Jiaotong University*, 2(7):220–222, 2002.

[82] Z.H. Hu and Y.G. Li. An empirical comparison of ensemble classification algorithms with support vector machines. *Proceeding of ICMLC2004 Conference*, 2004.

[83] H.C. Kim and S. Pang. Constructing support vector machine ensemble. *Pattern Recognition*, 36:2757–2767, 2003.

[84] Y.S. Dong and K.S. Han. A comparison of several ensemble methods for text categorization. *IEEE International Conference on Services Computing*, 2004.

[85] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[86] G. Lebrun, C. Charrier, and O. Lezoray. Tabu search model selection for svm. *International Journal of Neural Systems - Special Issue on Issue's Topic.*

[87] J. Ji, Z. Huang, C. Liu, X. Liu, and N. Zhong. An ant colony optimization algorithm for solving the multidimensional knapsack problems. *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 10–16, 2007.

[88] W. Xiong and C. Wang. Feature selection: A hybrid approach based on selfadaptive ant colony and support vector machine. *2008 International Conference on Computer Science and Software Engineering*, pages 751–754, 2008.

[89] Y.W. Chen and C.J. Lin. Combining svms with various feature selection strategies. 2005. URL http://www.csie.ntu.edu.tw/~cjlin/papers/features.pdf.

[90] H.W. Ressom, R.S. Varghese, S.K. Drake, G.L. Hortin, M. Abdel-Hamid, C.A. Loffredo, and R. Goldman. Peak selection from maldi-tof mass spectra using ant colony optimization. *Bioinformatics Advance Access*, 2007.

[91] J. Acevedo, S. Maldonado, S. Lafuente, H. Gomez, and P. Gil. Model selection for support vector machines using ant colony optimization in an electronic nose application. *Ant Colony Optimization and SWARM Intelligence*, 4150:468–475, 2006.

[94] G. Dolce, F. Riganello, M. Quintieri, A. Candelieri, and D. Conforti. Personal interaction in vegetative state: the mom effect. *Journal of Psychophysiology*, 22(3): 150–156, 2008.

[92] F. Riganello, M. Quintieri, A. Candelieri, D. Conforti, and G. Dolce. Heart rate response to music: an artificial intelligence study on healthy and brain injured subjects. *Journal of Psychophysiology*, 22(4):166–174, 2008.

[93] F. Riganello, V. Lagani, L. Pignolo, and A. Candelieri. Data mining approaches for the study of emotional response in healthy controls and traumatic brain injured patients: comparative analysis and validation. *In Proceedings of ICINCO 2009 - 6th International Conference on Informatics in Control, Automation and Robotics: 5th International Workshop on Artificial Neural Networks and Intelligent Information Processing (ANNIIP), Milan, Italy,July 2-5, 2009*, 2009.

[95] A. Candelieri, D. Conforti, F. Perticone, A. Sciacqua, K. Kawecka-Jaszcz, and K. Styczkiewicz. Early detection of decompensation conditions in heart failure patients by knowledge discovery: the heartfaid approaches. *In Proceedings of Computers in Cardiology, Bologna, Italy, September 14-17, 2008*, 35.

Ringraziamenti

Vorrei sfruttare solo poche righe per ringraziare le persone che mi hanno tenuto compagnia in molti momenti della mia vita, e per ultimi i tre anni di dottorato: la mia famiglia, Vale e gli amici tutti, specialmente Anto "Lucino" e Domenico "Doah". Un ringraziamento speciale anche ai miei 3 magnifici amici e colleghi Vincenzo, Rosita e Debora.
Infine, tutta la mia gratitudine ed il più affettuoso grazie vanno al Prof. Domenico Conforti per la sua fiducia, i suoi consigli e per essere (forse inconsapevolmente) uno splendido esempio di vita, professionale e umano.