UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Magistrale in Informatica

Apprendimento Simultaneo di Fuzzy Set

Relatore:
Prof. Dario Malchiodi
Correlatore:
Prof. Anna Maria Zanaboni

Tesi di Laurea di:
Luca CERMENATI
Matricola: 868106

Anno Accademico 2017/2018

Indice

Introduzione

Secondo i criteri classici di appartenenza a un insieme, è sempre possibile stabilire se un elemento ne è parte oppure no. D'altro canto, alcune classi di oggetti, per loro stessa natura, non possono essere definite tramite criteri d'appartenenza precisi e demarcati. In letteratura questi insiemi sono noti come insiemi fuzzy. L'obiettivo di questa tesi è presentare un metodo per l'apprendimento di tali classi di oggetti partendo da un insieme di dati e, in particolare, apprendere molteplici insiemi fuzzy in modo simultaneo. Il procedimento proposto è un'estensione e un adattamento dell'algoritmo di *support vector clustering*. Inoltre, nella costruzione di tale procedimento ci si avvale di lavori precedenti riguardanti l'apprendimento di un singolo insieme fuzzy. Una volta definito analiticamente, l'algoritmo di apprendimento simultaneo è stato implementato e validato attraverso una serie di esperimenti su un insieme di dati noto nell'ambito dell'apprendimento automatico. I risultati ottenuti sono stati in seguito confrontati con quelli di algoritmi noti, mostrando come l'algoritmo proposto risulti essere in media più accurato.
Il lavoro è presentato nella maniera seguente. Nel primo capitolo viene introdotto il concetto di insieme fuzzy e ne viene fornita una definizione formale presentandone proprietà, differenze e analogie con gli insiemi intesi come nel senso classico del termine. Nel secondo capitolo sono introdotti i concetti base riguardanti i problemi di classificazione e i diversi metodi di risoluzione riguardanti le support vector machine. Il secondo capitolo illustra anche il metodo di support vector clustering, basato sulle support vector machine, sul quale si poggia il metodo di apprendimento proposto. Nel terzo capitolo viene mostrato un approccio per l'apprendimento di un unico fuzzy set, unitamente a un esempio di utilizzo nel caso supervisionato. Infine, nel quarto e ultimo capitolo viene proposta la procedura per l'apprendimento di più fuzzy set, derivandola analiticamente e illustrandone alcuni esempi pratici. Vengono presentate nello specifico le metodologie di validazione e i risultati ottenuti durante la fase sperimentale, commentando gli esiti ottenuti. Segue quindi un paragrafo relativo alla complessità dell'algoritmo. Il lavoro termina con delle conclusioni in cui sono lasciati degli spunti per possibili estensioni future.

Capitolo 1

Insiemi fuzzy

1.1 Il concetto di fuzziness

La logica classica poggia su due valori di verità differenti e opposti, *vero* e *falso*. Ogni enunciato affrontabile secondo i criteri della logica classica deve essere una *proposizione*; deve essere cioè possibile dire con *certezza* se l'enunciato è vero o in alternativa se è falso. Quindi gli enunciati "Luca è alto 180 cm" e "3 è maggiore di 5" sono proposizioni, mentre l'enunciato "Milano è una bella città" non è una proposizione. Rappresentare un insieme in maniera *intensiva* significa scrivere un enunciato generalizzato che catturi una caratteristica comune a tutti e soli gli elementi appartenenti all'insieme che si vuole rappresentare. Per esempio $\{x \in \mathbb{R} \mid x > 0\}$ esprime la caratteristica di essere un numero reale e di essere maggiore di zero, ed è la rappresentazione intensiva dell'insieme dei numeri reali positivi. Preso un singolo elemento x, l'enunciato generale della rappresentazione intensiva diventa una proposizione. A stabilire se l'elemento considerato è parte oppure no dell'insieme è la veridicità della proposizione ottenuta. Ne deriva che il concetto di appartenenza a un insieme, così come quello di veridicità di un enunciato a cui è legato, è un concetto binario, può assumere cioè solo due valori distinti. Per comodità questi valori vengono codificati considerando gli elementi dell'insieme $\{0, 1\}$. La veridicità di un enunciato è quindi una funzione $v(\cdot)$ tale che

$$v(x) = \begin{cases} 0 & \text{se } x \text{ è falso,} \\ 1 & \text{se } x \text{ è vero.} \end{cases} \tag{1.1}$$

Analogamente l'appartenenza a un generico insieme A è una funzione μ_A tale che

$$\mu_A(x) = \begin{cases} 0 & \text{se } x \notin A, \\ 1 & \text{se } x \in A. \end{cases} \tag{1.2}$$

Una funzione di questo tipo viene chiamata *funzione di appartenenza* (o *membership function*) di A. L'utilizzo di proposizioni nella definizione di un

insieme porta a una separazione netta tra gli elementi che appartengono all'insieme e quelli che invece non vi appartengono. Tale separazione è dovuta alla *certezza* con cui è possibile stabilire, per definizione, il valore di verità di una proposizione. Chiamiamo questo tipo di insiemi *crisp*. Non tutti gli insiemi definibili hanno però dei criteri di appartenenza precisi di tipo binario. Alcuni insiemi per loro stessa natura hanno confini che non sono nettamente delineabili. Si considerino per esempio l' insieme delle "persone giovani" o quello dei "numeri reali molto più grandi di 1". Costringere questi insiemi in una visione binaria di appartenenza vorrebbe dire imporre un'età precisa passata la quale non si è più giovani e un punto sulla retta dei reali che faccia da spartiacque tra i numeri molto più grandi di uno e quelli solamente più grandi. In entrambi i casi una rappresentazione in termini binari non coglierebbe a pieno la vera natura dei due insiemi. Un *insieme fuzzy* è un insieme del tipo appena descritto, un insieme i cui confini risultano essere per natura sfocati e poco chiari. La rappresentazione di un insieme fuzzy è basata sull'uso di una logica non binaria (*logica fuzzy*) che permette a un enunciato di assumere valori di verità nell'intervallo $[0, 1]$, e conseguentemente di avere funzioni di appartenenza che restituiscano valori nello stesso intervallo. Quindi, per esempio, una possibile rappresentazione fuzzy per l'insieme A: "Numeri reali molto più grandi di 1" potrebbe essere

$$\mu_A(x) = \begin{cases} 0 & \text{se } x \leq 1, \\ \dfrac{x}{100} & \text{se } 1 < x \leq 100, \\ 1 & \text{se } x > 100, \end{cases} \tag{1.3}$$

che meglio rappresenta il concetto di *molto* più grande alla base della defi-

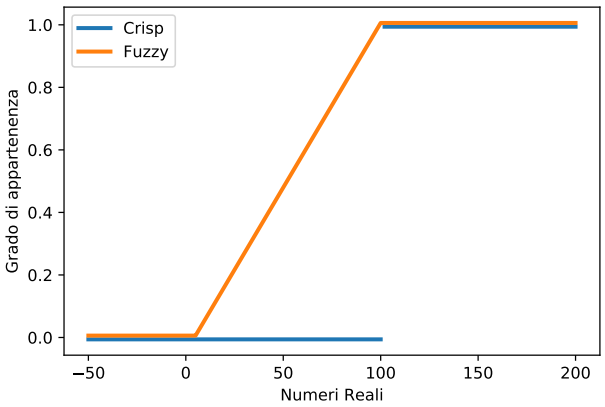
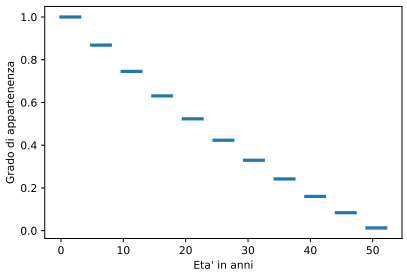


Figura 1.1: Grafico delle funzioni di appartenenza all'insieme "Numeri molto più grandi di 1" nelle versioni crisp e fuzzy

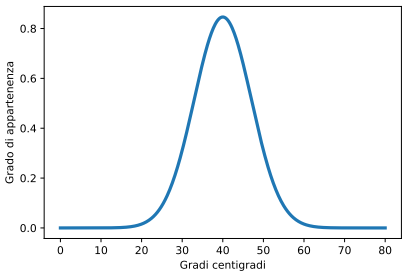
nizione dell'insieme A. La Figura 1.1 mostra una rappresentazione grafica di questa funzione di appartenenza, sovrapposta a un'analoga rappresentazione del concetto di appartenenza per un insieme *crisp*.

Nonostante l'appartenenza venga misurata con un numero compreso tra zero e uno, il concetto di appartenenza fuzzy non ha niente a che vedere con la probabilità che un elemento ha di appartenere a un certo insieme e la logica fuzzy non ha niente a che vedere con la probabilità che coinvolge variabili casuali e tratta di eventi e della loro possibilità di verificarsi. La *fuzziness* è il risultato dell'imprecisione e dell'assenza di criteri chiari e ben definiti di appartenenza a un insieme.
Pertanto un insieme fuzzy è una collezione di oggetti con un grado non binario di appartenenza. Tale insieme è caratterizzato da una funzione che assegna a ogni oggetto un grado di appartenenza tra zero e uno, come formalmente indicato nella definizione seguente [1].

Definizione 1 *Sia Ω un insieme di punti in uno spazio, il cui generico elemento di Ω è denotato usando il simbolo x. Un insieme fuzzy (o fuzzy set) $A \subseteq \Omega$ è caratterizzato da una funzione di appartenenza (o membership function) $\mu_A(\cdot)$ che associa a ogni punto $x \in \Omega$ un valore reale nell'intervallo $[0, 1]$, che rappresenta il grado di appartenenza di x a A.*



(a) Insieme delle persone giovani



(b) Insieme delle bevande tiepide

Figura 1.2: Alcuni esempi di fuzzy set con relativo grafico della funzione di appartenenza. Nella Figura 1.2(a) l'insieme delle persone giovani è caratterizzato da una funzione costante a tratti e decrescente. Nella Figura 1.2(b) l'insieme delle bevande tiepide è caratterizzato da una funzione continua a campana.

1.2 Operazioni tra insiemi fuzzy

Sia per un insieme fuzzy, sia per un insieme del tipo ordinario, il grado di appartenenza zero ha il significato di *non* appartenenza all'insieme e il grado di appartenenza 1 ha invece il significato di appartenenza. Gli insiemi fuzzy possono essere dunque visti come un'estensione degli insiemi crisp, mentre questi ultimi come un caso particolare di insiemi fuzzy la cui funzione di appartenenza è di tipo binario. Tutte le operazioni che si possono compiere con gli insiemi crisp sono estendibili agli insiemi fuzzy e tutte le definizioni fuzzy sono riconducibili a quelle crisp. Di seguito sono passate in rassegna le principali operazioni che coinvolgono degli insiemi, descrivendo come queste

possano essere estese agli insiemi fuzzy.

Insieme vuoto. Un insieme fuzzy A è vuoto, scritto $A = \varnothing$, se e solo se

$$\mu_A(x) = 0 \; \forall \; x \in \Omega. \tag{1.4}$$

Uguaglianza. Due insiemi fuzzy A e B sono uguali, scritto come $A = B$, se e solo se

$$\mu_A(x) = \mu_B(x) \; \forall \; x \in \Omega. \tag{1.5}$$

Sottoinsiemi. Un insieme fuzzy A è sottoinsieme di un insieme fuzzy B, scritto come $A \subseteq B$, se e solo se

$$\mu_A(x) \leq \mu_B(x) \; \forall \; x \in \Omega. \tag{1.6}$$

Complemento. Il complemento di un fuzzy set A è il fuzzy set $\bar{A}$, la cui funzione di appartenenza $\mu_{\bar{A}}$ è tale che

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \; \forall \; x \in \Omega. \tag{1.7}$$

Unione. L'unione tra un insieme fuzzy A e un insieme fuzzy B è un insieme fuzzy C, scritto $C = A \cup B$, la cui funzione di appartenenza μ_C è tale che

$$\mu_C(x) = \max[\mu_A(x), \mu_B(x)] \; \forall \; x \in \Omega \tag{1.8}$$

Intersezione. L'intersezione tra un insieme fuzzy A e un insieme fuzzy B è un insieme fuzzy C, scritto $C = A \cap B$, la cui funzione di appartenenza μ_C è tale che

$$\mu_C(x) = \min[f_A(x), f_B(x)] \; \forall \; x \in \Omega. \tag{1.9}$$

Disgiunzione. Due insiemi fuzzy A e B sono disgiunti se

$$A \cap B = \varnothing. \tag{1.10}$$

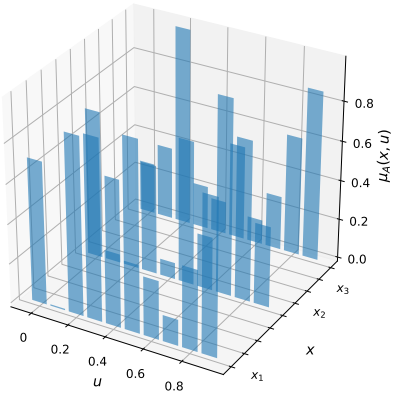
Va notato come queste estensioni non siano necessariamente uniche. Per esempio l'unione $A \cup B$ si può definire come il *più piccolo* insieme fuzzy C che contiene sia A che B. Viceversa l'intersezione $A \cap B$ può essere definita come il *più grande* insieme fuzzy C che è contenuto sia in A che in B. In entrambi i casi gli operatori $\cup$ e $\cap$ sono associativi.

1.3 Insiemi fuzzy di tipo 2

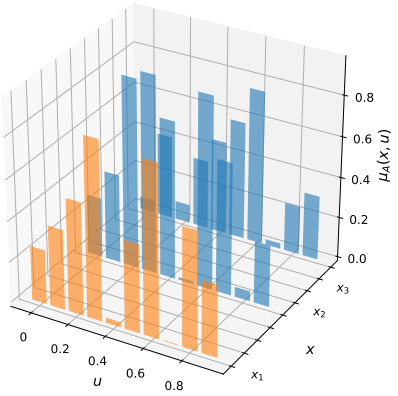
Come visto nel paragrafo precedente, un fuzzy set modella un insieme i cui confini non sono ben definiti o sono incerti. Non vi è nessuno tipo di incertezza invece riguardo al grado di appartenenza $\mu_A(x)$ di un elemento x a un insieme A. Gli insiemi fuzzy caratterizzati da una funzione di appartenenza *non* incerta sono definiti insiemi fuzzy di tipo 1. Al contrario, gli insiemi fuzzy di tipo 2 ammettono funzioni di appartenenza incerte [2].

Definizione 1 *Un insieme fuzzy di tipo 2 A è caratterizzato da una funzione di appartenenza $\mu_A(x, u)$, con $x \in \Omega$ e $u \in J_x \subseteq [0, 1]$. Come per i fuzzy set del tipo 1 vale sempre $0 \leq \mu_A(x, u) \leq 1$.*

Il valore u equivale al grado di appartenenza $\mu_A(x)$ di x all'insieme fuzzy di tipo 1 A. Il risultato di una funzione di appartenenza di tipo 2 è l'incertezza su un un certo grado di membership u per un dato elemento x. Quando questa incertezza viene meno un insieme fuzzy di tipo 2 si riduce a un insieme fuzzy di tipo 1. Viceversa gli insiemi fuzzy di tipo 2 sono un'estensione degli insiemi fuzzy di tipo 1 con l'aggiunta di un grado di incertezza.



(a) Grafico di una funzione di appartenenza di tipo 2



(b) A ogni valore $x \in \Omega$ corrisponde un piano i cui assi sono u e $\mu_A(x, u)$. In evidenza il piano corrispondente a x_1

Figura 1.3: Insiemi fuzzy di tipo 2.

Capitolo 2

Le support vector machine

2.1 Classificazione

Classificare significa assegnare oggetti a classi secondo criteri di affinità. Un problema di classificazione consiste nella costruzione di una regola $f : \Omega \to Y$ che associa gli elementi del dominio Ω alle *etichette* appartenenti al codominio Y dove ogni diversa etichetta individua una diversa classe. L'insieme Ω tipicamente è un sottoinsieme $\Omega \subseteq \mathbb{R}^d$. Il generico elemento di Ω è quindi un punto d-dimensionale. Ciascuna delle d dimensioni equivale a una specifica *feature* della famiglia di oggetti Ω. Nell'ambito del machine-learning il processo di costruzione della regola f è detto *apprendimento*. Durante la fase di apprendimento , un algoritmo visiona un insieme di esempi $T = \{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$, dove la generica coppia (x_i, y_i), con $x_i \in X \subseteq \Omega$ e $y_i \in Y$, è un esempio di classificazione corretta di x_i con l'etichetta y_i, e ne estrae un criterio di classificazione generale per tutti i punti appartenenti a Ω, cioè la funzione $f(\cdot)$. L'insieme T prende il nome di *training set*.
La notazione x_i indica il primo elemento della i-esima coppia appartenente a un training set. Mentre la notazione $x_i^{(j)}$ indica la j-esima componente del vettore x_i. Quando la cardinalità dell'insieme di arrivo è $|Y| = 2$, si parla di classificazione binaria. Per comodità di notazione e di calcolo da qui in poi verranno presentati esempi di classificazione binaria in cui l'insieme Y delle possibili etichette è $Y = \{-1, 1\}$.

2.1.1 Percettrone

Il percettrone [3] è un classificatore binario a soglia che assegna un oggetto $x \in \Omega$ alla classe ottenuta utilizzando la seguente regola:

$$f_P(x) = \begin{cases} +1 & se\ w \cdot x \geq \theta, \\ -1 & se\ w \cdot x < \theta, \end{cases} \tag{2.1}$$

dove w è un vettore di pesi delle stesse dimensioni del vettore di input x, θ è la soglia discriminante nella classificazione di un punto, ilsimbolo $\cdot$ indica il prodotto scalare. Se la somma delle componenti del vettore input x pesate secondo quelle del vettore w supera una certa soglia θ, il percettrone dà un responso *positivo*, altrimenti *negativo*. L'iperpiano di equazione $w \cdot x = \theta$ separa in Ω la classe positiva da quella negativa. La classificazione tramite percettrone è quindi applicabile a classi *linearmente separabili* nello spazio di definizione. La fase di apprendimento dell' iperpiano separatore $w^* \cdot x = \theta$ consiste nell'aggiustare iterativamente i valori di w fino a quando non si ottiene un iperpiano separatore corretto per il training set visionato, secondo una procedura simile a quella descritta nell'Algoritmo 1. Si può dimostrare [3] che quando le due classi sono linearmente separabili,

Algoritmo 1 Ricerca dell'iperpiano separatore

1: inizializzare w con un vettore casuale
2: **while** esistono errori di classificazione **do**
3: $\quad i = 0$
4: $\quad$ **while** $i < |T|$ **do**
5: $\quad\quad y' = w \cdot x_i - \theta$
6: $\quad\quad$ **if** $y' \cdot y < 0$ **then** $w = w + \eta y x_i$
7: $\quad\quad$ **end if**
8: $\quad$ **end while**
9: **end while**

il Punto 9. viene superato dopo un numero finito di iterazioni. In tal caso, l'iperpiano separatore è individuato da $w^* \cdot x = \theta$, dove w^* è il valore di w al termine dell'esecuzione dell'Algoitmo 1. In presenza di classi *non* linearmente separabili il precedente algoritmo rimane bloccato in un loop infinito. L'Istruzione 5. esegue l'aggiornamento di w, facendo in modo che il nuovo w risulti spostato verso il vettore x_i che è mal classificato in quel momento. Il parametro $\eta > 0$ definisce la misura dello spostamento e di conseguenza la velocità con cui w si avvicina alla soluzione. Un η troppo grande potrebbe portare a una situazione di loop infinito in cui w continua ad aggiornarsi tra due valori, uno più grande, uno più piccolo della soluzione, o anche a far divergere w. Nell'Algoritmo 1 la soglia θ è trattata come costante. È possibile permettere alla soglia di variare eliminandola di fatto dalla formalizzazione e aggiungendo una variabile in coda, come nuova dimensione, al vettore w. Aggiungendo agli esempi x_i con lo stesso metodo il peso -1 il calcolo di y' risulta analogo a quello del punto 3 e la soglia è in grado di variare come ogni altra singola componente di w. Accettando di avere a che fare con classi non separabili l'algoritmo può essere modificato permettendo lo stop secondo alcuni criteri, per esempio eseguendo un numero fissato di cicli, oppure fermandosi quando il numero di classificazioni errate smette di variare al variare di w.

2.1.2 Macchine a vettori di supporto

Una macchina a vettori di supporto [4], *support vector machine*, o SVM in breve, ha come scopo non solo quello di trovare un iperpiano $w \cdot x + b = 0$ che separi linearmente i dati, ma anche di fare in modo che l'iperpiano minimizzi l'errore durante la fase di classificazione di punti al di fuori dell'insieme di training. Questo iperpiano è quello posizionato alla stessa distanza da entrambe le classi. La distanza di una classe da un piano equivale alla minima distanza punto-piano misurata per ogni punto appartenente alla classe. Questa distanza viene definita *margine*. Utilizzare l'iperpiano equidistante dalle classi per la discriminazione tra le due permette di massimizzare il margine e ottenere così una migliore generalizzazione, cioè una migliore capacità di classificare correttamente anche punti non presenti nel training set. In Figura 2.1 la rappresentazione grafica di quanto descritto finora.

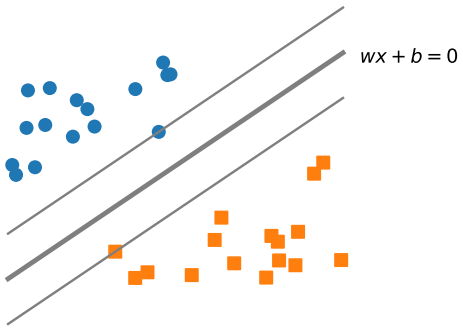


Figura 2.1: Iperpiano separatore per due classi linearmente separabili in due dimensioni. Il margine equvale alla distanza tra la retta di equazione $wx + b = 0$ e le due rette più sottili.

La fase di apprendimento di SVM consiste nella massimizzazione del margine γ, variando i parametri w e b, soggetta al vincolo che per ogni punto x_i valga

$$y_i(w \cdot x_i + b) \geq \gamma. \tag{2.2}$$

Indicando con w^* e b^* le soluzioni del precedente problema di massimizzazione sono individuati

$w \cdot x + b = 0$, l'iperpiano separatore.

$w \cdot x + b = \gamma$, il confine della classe positiva.

$w \cdot x + b = -\gamma$, il confine della classe negativa.

I punti che soddisfano l'equazione $w^* \cdot x + b^* = \pm\gamma$, si trovano esattamente a distanza γ dal separatore e si trovano sul confine delle loro classi. Tali

punti prendono il nome di *support vector* o *vettori di supporto*. La Figura 2.2 mette in evidenza i vettori di supporto per due classi linearmente separabili. Scelta una coppia (w^*, b^*) di soluzioni, è sempre possibile trovare

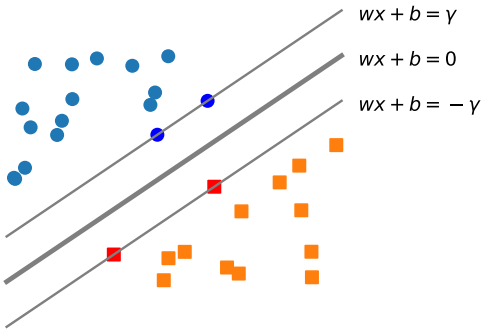


Figura 2.2: Vettori di supporto per due classi di punti linearmente separabili.

una coppia (kw^*, kb^*), per $k > 1$, anch'essa soluzione, per cui è sempre vero che $k\gamma > \gamma$. In altre parole, γ non può essere massimizzato. La soluzione a questo problema consiste nella normalizzazione del vettore w costringendo γ a essere multiplo del vettore unità $\frac{w}{\| w \|}$. Questo cambia il problema di ottimizzazione nella minimizzazione di $\| w \|$ soggetta al vincolo che per ogni punto x_i valga

$$y_i(w \cdot x_i + b) \geq 1. \tag{2.3}$$

Una volta determinati tali valori ottimali, la classificazione viene eseguita secondo la regola

$$f_{\text{SVC}}(x) = \begin{cases} +1 & se\ w^* \cdot x + b^* \geq 0, \\ -1 & se\ w^* \cdot x + b^* < 0. \end{cases} \tag{2.4}$$

Va notato che il problema di minimizzazione di $\| w \|$ e quello di una qualsiasi funzione monotona di $\| w \|$ hanno le stesse soluzioni. Per comodità di derivazione e risoluzione normalmente si sceglie di costruire il problema di minimizzazione attorno alla funzione $f(\| w \|) = \frac{1}{2} \| w \|^2$, trasformando il problema nel minimizzare $\frac{1}{2} \| w \|^2$ soggetta ai vincoli $y_i(w \cdot x_i + b) \geq 1 \; \forall \, x_i \in X$. Scrivere la lagrangiana

$$L = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i + b - 1)], \tag{2.5}$$

dove ogni $\alpha_i \geq 0$ è un moltiplicatore lagrangiano, permette di inglobare i vincoli nel problema. Differenziando L rispetto a w, e rispetto a b poi, e ponendo tali derivate uguali a zero, si ottiene

$$\frac{\partial L}{\partial w} = 0 \Longrightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i, \tag{2.6}$$

$$\frac{\partial L}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.7}$$

Questi risultati, sostituiti in (2.5), portano a quello che è definito *problema duale* [5]: massimizzare, variando le nuove variabili α_i,

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.8}$$

soggetta ai vincoli

$$\alpha_i \geq 0 \; \forall i \in \{1, \cdots n\}, \tag{2.9}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.10}$$

Inoltre, per le condizioni di Karush-Kuhn-Tucker [6] (condizioni KKT), deve valere per ogni $i \in \{1, \cdots, n\}$

$$\alpha_i [y_i (w \cdot + b - 1)] = 0. \tag{2.11}$$

Risulta quindi che per le soluzioni $\alpha_i^* > 0$ del problema duale è soddisfatta l'uguaglianza $y_i(w \cdot x_i + b) = 1$. Di conseguenza x_i è un vettore di supporto e w^* è ricavabile da (2.6), come combinazione lineare dei vettori di supporto. Mentre b^* è la soluzione dell'equazione $y_i(w^* \cdot x_s + b) = 1$ dove x_s è uno tra i vettori di supporto individuati.

2.1.3 Kernel Trick

Quello di utilizzare uno, o più, piani separatori è un buon metodo di classificazione in presenza di dati linearmente separabili. Al contrario, in presenza di dati non linearmente separabili, qualsiasi scelta per la coppia (w^*, b^*) produce un iperpiano che mal classifica almeno un elemento del training set. La Figura 2.3 rappresenta graficamente una di queste situazioni.
Una delle possibili soluzioni è quella di trasformare i dati in modo che risultino essere linearmente separabili utilizzando il *kernel trick*. L'idea che sta dietro al *trucco* è quella che punti d-dimensionali, non linearmente separabili in d dimensioni, potrebbero esserlo se trasportati in uno spazio a maggiore dimensionalità. Una funzione $\phi : \mathbb{R}^d \to \mathbb{R}^D$, per $D > d$, che trasporta punti in uno spazio a maggiore dimensionalità aggiungendo le feature mancanti come funzione di quelle originali è chiamata *mapping*. Il codominio di un mapping invece prende il nome di *spazio delle feature*. Per sortire l'effetto

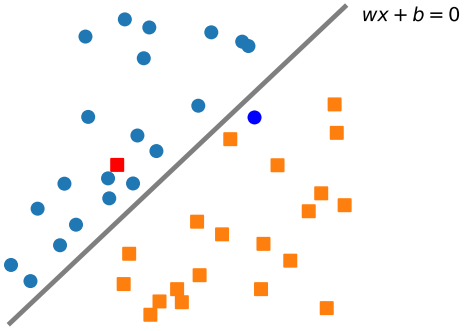


Figura 2.3: Esempio in due dimensioni di classi non linearmente separabili. In evidenza i punti male classificati.

di produrre dati linearmente separabili un mapping deve poter proiettare in porzioni differenti dello spazio punti appartenenti a differenti classi.

Esempio 1. Dato il training set $T_1 = \{(1,a),(-2,a),(3,b),(-3,b),(-4,b),(-1,a),(4,b),(2,a)\}$ dove $X = \{x \in \mathbb{N} | -5 < x < 5\}$ e $Y = \{a,b\}$. Le due differenti classi risultano non linearmente separabili sulla retta dei reali [Figura 2.4(b)]. Il mapping $\phi_1(x) = (x, x^2)$ proietta verso l'alto gli $x \in X$ con modulo maggiore, rendendo il training set linearmente separabile nelle due dimensioni (Figura 2.4(b)).

Esempio 2. Dato il training set $T_2 = \{[(0;3),a],[(3;0),a],[(1;2),b],(2;1),b],(3/2;0),a],[(0;3/2),a],[(1;1),b],[(2;2),b],[(0;0),a],[(3/2;3/2)]\}$, con $X \subset \mathbb{R}^2$. Le classi $a, b \in Y$ non sono linearmente separabili [Figura 2.4(c)]. La trasformazione $\phi_2(x,y) = (x,y,xy)$ produce valori bassi nella terza dimensione per i punti etichettati a, valori alti per i punti etichettati b, rendendo l'immagine di X linearmente separabile nello spazio tridimensionale [Figura 2.4(d)].

In questo caso la regola di classificazione diventa quindi

$$f_K(x) = \begin{cases} +1 & se\ w^* \cdot \phi(x) + b^* \geq 0, \\ -1 & se\ w^* \cdot \phi(x) + b^* < 0. \end{cases} \tag{2.12}$$

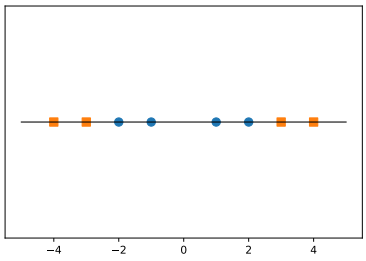
La denominazione *kernel trick* deriva dal fatto che il trucco è utilizzato in coppia con funzioni denominate *kernel function* [7]. Considerato un mapping ϕ, una kernel function K è definita in modo tale che sia sempre valido

$$K(x_i, x_j) = \phi(x_i) \cdot \phi x_j. \tag{2.13}$$

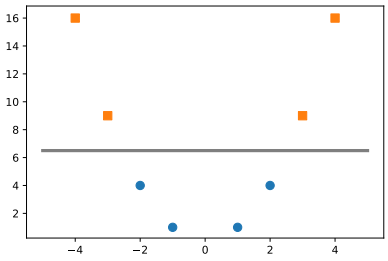
In questo modo, conoscendo K, è possibile calcolare il prodotto scalare tra due elementi nello spazio delle feature senza dover effettivamente calcolare i

mapping corrispondenti. Così facendo la (2.8) può essere riscritta in questo modo:

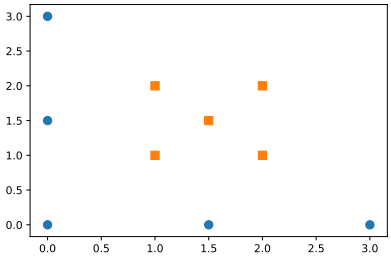
$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \tag{2.14}$$



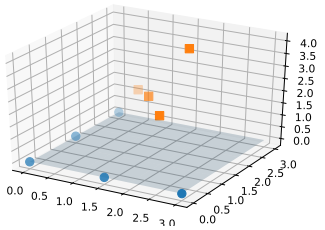
(a) Training set T_1 in $\mathbb{R}$



(b) Training set T_2 mappato in $\mathbb{R}^2$ da ϕ_1



(c) Training set T_2 in $\mathbb{R}^2$



(d) Training set T_2 mappato in $\mathbb{R}^3$ da ϕ_2

Figura 2.4: Training set degli esempi 1 e 2. A sinistra nello spazio originale, a destra dopo la trasformazione in più dimensioni.

Nei capitoli successivi ogniqualvolta si fa riferimento a una funzione kernel si intende una funzione kernel di tipo gaussiano [8] così definita:

$$\phi(x) = e^{-\frac{x^2}{2\sigma^2}} [1, \sqrt{\frac{1}{1!\sigma^2}} x, \sqrt{\frac{1}{2!\sigma^4}} x^2, \sqrt{\frac{1}{3!\sigma^6}} x^3, \cdots]^T. \tag{2.15}$$

La particolarità del kernel gaussiano è quella di mappare punti in un insieme con infinite dimensioni. Il risultato del prodotto scalare $\phi(x_i) \cdot \phi(x_j)$ risulta invece essere

$$K(x_i, x_j) = e^{-\frac{\| x_i - x_j \|^2}{2\sigma^2}}. \tag{2.16}$$

2.1.4 Dati non linearmente separabili

L'utilizzo del kernel trick non è sempre garanzia di separazione lineare delle classi. Inoltre in alcune situazioni lasciare che i dati siano non separabili è

funzionale alla risoluzione del problema di classificazione. Anche in questo caso nessuna coppia (w^*, b^*) individua un iperpiano che non commette errori di classificazione. Non potendo essere eliminati totalmente, gli errori di classificazione diventano parte integrante della ricerca del criterio di classificazione. Il migliore iperpiano apprendibile da un training set diventa allora quello che commette il minor numero di errori, o gli errori più veniali, e si mantiene sufficientemente lontano dai confini delle classi. Un iperpiano incorre in una penalità ogniqualvolta classifichi in modo errato un punto o gli si avvicini a una distanza minore di un certo γ. I punti classificati correttamente ma troppo vicini al piano vengono definiti *outlier*(Figura 2.5). Ogni penalità è pesata in base alla distanza del punto dall'iperpiano. Il piano che minimizza l'ammontare $l(w, b)$ della penalità è quello poi utilizzato per classificare i punti al di fuori del training set.

$$l(w,b) = \frac{1}{2}\sum_{j=1}^{d} w^{(j)^2} + C\sum_{i=1}^{n} \max[0, 1 - y_i(\sum_{j=1}^{d} w^{(j)} x_i^{(j)} + b)]. \tag{2.17}$$

Il primo termine di $l(w, b)$, al fine di normalizzare $\| w \|^2$ come visto nel Paragrafo 2.1.2, favorisce w composti da piccole componenti. Il parametro di *tradeoff* $C > 0$ indica quanto l'ammonatare della penalità pesa rispetto al precedente addendo. Valori bassi di C ammettono più errori di classificazione ma allargano i margini tra iperpiano e classi, mentre valori alti di C ammettono meno errori di classificazione ma diminuiscono i margini.

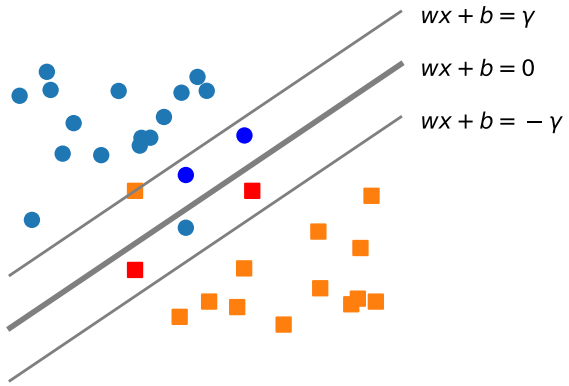


Figura 2.5: In evidenza gli outlier per un insieme di punti bidimensionali. Nel grafico sono presenti anche due errori di classificazione.

Anche in questa situazione il criterio di classificazione è dato dalla

$$f_{\text{SVC}}(x) = \begin{cases} +1 & se\ w^* \cdot x + b^* \geq 0, \\ -1 & se\ w^* \cdot x + b^* < 0. \end{cases} \tag{2.18}$$

che differisce dalla 2.4 per il metodo di training dell'iperpiano $w^* \cdot x + b^* = 0$. È utile notare che allo stesso modo che per (2.5) è possibile ricavare il problema duale alla minimizzazione di $l(w, b)$, e cioè massimizzare

$$l_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.19}$$

soggetta ai vincoli

$$0 \leq \alpha_i \leq C \; \forall \; i \in \{1, \cdots n\}. \tag{2.20}$$

Infine, gli approcci proposti finora possono essere mescolati, come è il caso del problema denominato 1-SVM affrontato nel prossimo paragrafo.

2.2 1-SVM

1-SVM è un particolare caso di classificazione mediante macchine a vettori di supporto in cui il numero di classi presenti nel problema è pari a uno. Distribuire punti all'interno di una singola classe non significa costruire una funzione che restituisce l'unica possibile etichetta per ogni possibile valore di input, ma racchiudere all'interno di una porzione di spazio più piccola possibile il maggior numero di punti di un training set, ammettendo alcuni outlier, come mostrato in Figura 2.6. Ciò permette successivamente di rilevare per esempio dei dati anomali. Formalmente, dato un insieme di oggetti $X = \{x_1, x_2, \cdots, x_n\}$, $X \subseteq \mathbb{R}^d$, e un mapping $\phi : \mathbb{R}^d \to \mathbb{R}^D$, con $D > d$, il problema di classificazione 1-SVM si riconduce alla ricerca nello spazio $\mathbb{R}^D$ della più piccola (*iper*)sfera di raggio R e centro a che racchiuda in sé la maggior parte dei punti appartenenti all'insieme X. Ciò equivale a minimizzare R sotto il vincolo che per ogni $x_i \in X$ valga

$$\| \phi(x_i) - a \|^2 \leq R^2 + \xi_i, \tag{2.21}$$

dove $\xi_i > 0 \; \forall \; i \in \{1, \cdots, n\}$ sono variabil slack. Analogamente a quanto visto nel Paragrafo 2.1.4 il problema di minimizzazione viene risolto introducendo una funzione lagrangiana:

$$L = R^2 - \sum_{i=1}^{n} (R^2 + \xi_i - \| \phi(x_i) - a \|^2)\beta_i - \sum_{i=1}^{n} \xi_i \mu_i + C \sum_{i=1}^{n} \xi_i. \tag{2.22}$$

Il primo addendo regola la misura del raggio, la prima sommatoria l'ampiezza dei margini tra la superficie della sfera e gli outlier, il termine $\sum_{i=1}^{n} \xi_i$ è un termine di penalità pesato per il parametro C. Mentre β_i e μ_i sono moltiplicatori lagrangiani. Ponendo uguale a zero la derivata di L questa volta rispetto a R, a e xi_i si ottiene

$$\sum_{i=1}^{n} \beta_i = 1, \tag{2.23}$$

$$a = \sum_{i=1}^{n} \beta_i \phi(x_i), \tag{2.24}$$

$$\beta_i = C - \mu_i. \tag{2.25}$$

Inoltre devono valere per ogni $i \in \{1, \cdots, n\}$ le condizioni KKT

$$\xi_i \mu_i = 0, \tag{2.26}$$

$$(R^2 + \xi_i - \| \phi(x_i) - a \|^2 \beta_i = 0). \tag{2.27}$$

Segue dalla (2.27) che l'immagine di un punto x_i a cui corrispondono $\xi_i > 0$ e $\beta_i > 0$ risiede al di fuori della sfera. Inoltre, essendo $\xi_i = 0$, per la condizione (2.26), al punto x_i deve anche corrispondere μ_i tale che $\mu_i = 0$. Conseguentemente, per la (2.25), risulta $\beta_i = C$. Se invece, β_i è tale che $0 < \beta_i < C$, per la (2.27), gli x_i corrispondenti sono mappati sulla superficie della sfera. Da quanto appena dimostrato consegue che ogni β_i^*, soluzione del problema di ottimizzazione, permette di classificare il corrispondente oggetto x_i nel modo descritto di seguito.

- Se $\beta_i^* = 0$ l'immagine di x_i secondo ϕ si trova all'interno della sfera.
- Se $0 < \beta_i^* < C$ l'immagine di x_i secondo ϕ si trova sulla superficie della sfera e x_i è un vettore di supporto.
- Se $\beta_i^* = C$ l'immagine di x_i è mappata al di fuori della sfera e x_i è considerato un outlier.

Successivamente, in modo simile a come visto per (2.8) e per (2.19), è possibile riscrivere il problema in forma duale di Wolfe [9]: dal problema vengono eliminate le variabili R, a ed μ_i, ottenendo il nuovo obiettivo

$$W = \beta_i \sum_{i=1}^{n} \phi(x_i)^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j \phi(x_i) \cdot \phi(x_j), \tag{2.28}$$

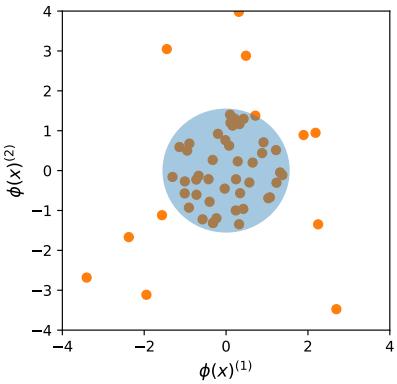


Figura 2.6: Esempio di sfera soluzione in uno spazio bidimensionale.

da ottimizzare rispettando i vincoli

$$0 < \beta_i \leq C \; \forall \; i \in \{1, \cdots, n\}. \tag{2.29}$$

Infine il prodotto scalare $\phi(x_i)$ può essere riscritto mediante l'utilizzo di una *kernel function*, ottenendo

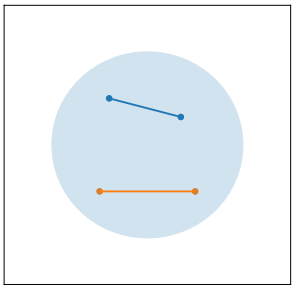
$$W = \sum_{i=1}^{n} \beta_i K(x_i, x_i) - \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j K(x_i, x_j). \tag{2.30}$$

A classificazione avvenuta il raggio R risulta essere uguale alla distanza dal centro di uno dei vettori di supporto individuati. Il quadrato della distanza di un generico punto x dal centro della sfera invece può essere calcolata in virtù di (2.24) mediante

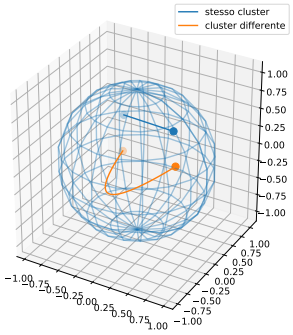
$$R^2(x) = K(x, x) - 2 \sum_{i=1}^{n} \beta_i K(x_i, x) + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j K(x_i, x_j). \tag{2.31}$$

2.3 Clustering

È possibile utilizzare 1-SVM per la risoluzione di problemi di clustering. Dopo aver applicato il procedimento descritto nel paragrafo precedente a un insieme di punti $a \in A \subseteq \mathbb{R}^d$, si dispone di una sfera che ingloba la maggior parte delle immagini $\phi(a)$ nello spazio delle feature e di una funzione $R^2(\cdot)$ che permette di calcolare il quadrato della distanza di una generica immagine $\phi(x)$ dal centro della sfera che rende così noto anche il raggio R. Conoscendo queste informazioni è possibile costruire un metodo per la divisione dei punti $a \in A$ in diversi cluster [10]. Due punti sono considerati appartenere allo stesso cluster se ogni punto del segmento che li unisce è mappato da ϕ all'interno della sfera, cioè risulti $R^2(p) \leq R^2$ per ogni punto p del segmento. Costruendo un grafo avente come nodi tutti i possibili $a \in A$ e tracciando un arco tra il nodo i ed il nodo j solo se il segmento che congiunge a_i con a_j è interamente mappato all'interno della sfera, le diverse componenti connesse del grafo corrispondono ai vari cluster in cui A è diviso. Questo metodo di divisione in cluster mediante 1-SVM è definito *support vector clustering*.



(a) Due coppie di punti collegate da un segmento in uno spazio bidimensionale.



(b) Coppie di punti collegate da un segmento in uno spazio tridimensionale.

Figura 2.7: Metodo di clustering 1-SVM. In Figura 2.7(a) due coppie di punti in uno spazio bidimensionale. In Figura 2.7(b) le stesse coppie di punti mappate in uno spazio tridimensionale. Si noti come l'immagine del segmento congiungente i due punti non appartenenti allo stesso cluster esca dalla sfera soluzione di 1-SVM.

Capitolo 3

Apprendimento di insiemi fuzzy

Come visto in precedenza, un insieme fuzzy è definito dalla sua funzione di appartenenza. Un algortimo di apprendimento per insiemi fuzzy di conseguenza deve essere costruito in modo tale che l'elaborazione delle informazioni presenti nel training set conduca alla costruzione di una funzione di appartenenza. Sia $X = \{x_1, x_2, \cdots, x_n\} \subset \Omega$ un insieme di punti d-dimensionali e $U = \{\mu_1, \mu_2, \cdots, \mu_n\}$ un insieme di gradi di appartenenza a un insieme fuzzy ignoto A, tale che μ_i è il grado di appartenenza di x_i a A. Apprendere l'insieme fuzzy A significa costruirne la funzione di appartenenza $\mu_A(\cdot)$ visionando gli insiemi di training X e U. Questo problema [11] consiste in due fasi: la ricerca dei confini dell'insieme fuzzy e la costruzione della sua funzione di appartenenza.

3.1 Ricerca dei confini

Come visto in precedenza, i confini di un insieme fuzzy non sono una linea di demarcazione netta tra i punti appartenenti all'insieme e quelli che invece non vi appartengono. La ricerca dei confini di un insieme fuzzy A consiste in realtà nella ricerca di un sottoinsieme $X_c \subseteq X$ a cui assegnare la proprietà $x \in X_c \rightarrow \mu_A(x) = 1$. L'insieme X_c così definito prende il nome di *core*. L'insieme fuzzy che ne risulta è quindi composto da un insieme di punti racchiusi all'interno di confini netti e da altri che invece giacciono al di fuori di essi, come è il caso degli outlier dell'algoritmo 1-SVM. Il problema (2.21) di ricerca della più piccola sfera di raggio R e centro a è modificato in modo tale da racchiudere il *core* dell'insieme fuzzy A all'interno della sfera tenendo conto del grado di appartenenza di ciascun punto del training set.

Il problema modificato consiste nel minimizzare

$$R^2 + C\sum_{i=1}^{n}(\xi_i + \tau_i) \tag{3.1}$$

soggetta ai vincoli

$$\mu_i \parallel \phi(x_i) - a \parallel \leq \mu_i R + \xi_i, \tag{3.2}$$

$$(1-\mu_i) \parallel \phi(x_i) - a \parallel \geq (1-\mu_i) R + \tau_i, \tag{3.3}$$

$$\xi_i \geq 0,\ \tau_i \geq 0, \tag{3.4}$$

che devono valere per ogni $i \in \{1, \cdots, n\}$. Il parametro di tradeoff C analogamente a quanto visto finora indica il peso delle penalità rispetto alla minimizzazione del raggio. Le variabili slack ξ_i quantificano una penalità per aver escluso dalla sfera un punto appartenente al core. Al contrario, le variabili slack τ_i quantificano una penalità per aver incluso all'interno della sfera un punto che non fa parte del core. Il primo tipo di penalità è pesata per il grado di appartenenza μ_i. Questo significa che escludere dalla sfera un punto x_i tale che $\mu_i = 1$ comporta una penalità completa, escluderne uno tale che $\mu_i = 0$ non comporta penalità: la gravità dell'errore è direttamente proporzionale al grado di appartenenza del punto. Il secondo tipo di penalità è invece inversamente proporzionale al grado di appartenenza del punto. Quindi per $\mu_i = 1$ i vincoli risultano identici a quelli di (2.21) e così il loro scopo, quello di confinare $\phi(x_i)$ all'interno della sfera. Al contrario con $\mu_i = 0$ i vincoli impongono di relegare x_i al di fuori di essa. Nella forma duale di Wolfe il problema si presenta come la massimizzazione di

$$\begin{aligned}&\sum_{i=1}^{m}(\alpha_i\mu_i - \beta_i(1-\mu_i))K(x_i,x_i) + \\ &-\sum_{i,j=1}^{m}(\alpha_i\mu_i - \beta_i(1-\mu_i))(\alpha_j\mu_j - \beta_j(1-\mu_j))K(x_i,x_j)\end{aligned} \tag{3.5}$$

soggetta ai vincoli

$$\sum_{i=1}^{m}(\alpha_i\mu_i - \beta_i(1-\mu_i)) = 1, \tag{3.6}$$

$$0 \leq \alpha_i, \beta_i \leq 1, \tag{3.7}$$

dove $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ è la *kernel function* utilizzata. Le soluzioni α_i, β_i del problema duale forniscono le seguenti informazioni sui punti $x_i \in X$:

- Se $\alpha_i = 0$, $\phi(x_i)$ si trova all'interno o sulla superficie della sfera.
- Se $\alpha_i = C$, $\phi(x_i)$ non si trova all'interno della sfera.
- Se $\beta_i = 0$, $\phi(xi_i)$ non si trova all'interno della sfera.

- Se $\beta_i = C$, $\phi(x_i)$ non si trova all'interno della sfera.

I vettori di supporto risultano essere gli x_i tali che $\alpha(x_i) = 0$ e $\beta_i = 0$ oppure $\beta_i = C$. Infine, analogamente a come visto per (2.31), il quadrato della distanza dal centro per l'immagine di un generico $x \in \Omega$ risulta essere

$$\begin{aligned} R^2(x) = K(x,x) - 2\sum_{i=1}^{m}(\alpha_i\mu_i - \beta_i(1-\mu_i))K(x,x_i) + \\ + \sum_{i,j=1}^{m}(\alpha_i\mu_i - \beta_i(1-\mu_i))(\alpha_j\mu_j - \beta_j(1-\mu_j))K(x_i,x_j). \end{aligned} \tag{3.8}$$

La superficie della sfera S soluzione del procedimento appena descritto delimita i confini del core dell'insieme fuzzy A nel codominio della funzione $\phi(\cdot)$.

3.2 Costruire la funzione di appartenenza

Una volta definito il core dell'insieme A occorre costruirne la funzione di appartenenza $\mu_A(\cdot)$ tenendo conto del fatto che, come tutte le funzioni di appartenenza a un insieme, deve restituire valori nell'intervallo $[0,1]$. Il procedimento presentato nel Paragrafo 3.1, grazie ai vincoli (3.2) e (3.3), rispecchia una certa incertezza nei confini dell'insieme A permettendo al raggio R della sfera S di variare in un intervallo del tipo $[R-\tau, R+\xi]$. Una volta risolto il problema di minimizzazione, R assume però un valore ben preciso, rendendo la superficie della sfera un limite crisp nella separazione tra il core e il resto dell'insieme. Occorre eseguire quindi un processo di *fuzzificazione* dell'insieme A. Questo processo è basato sull'ipotesi che il valore restituito dalla funzione di appartenenza $\mu_A(\cdot)$ dipenda esclusivamente dalla distanza $R^2(x)$ dal centro della sfera S. Ai punti mappati all'interno della sfera dalla funzione $\phi(\cdot)$ viene assegnato grado di membership pari a 1, mentre ai punti mappati al di fuori della sfera viene assegnato un valore compreso tra uno e zero, decrescente in funzione della distanza $R^2(\cdot)$. Ad esempio, sia

$$\varphi(x) = \frac{M - R^2(x)}{M - R^2}, \tag{3.9}$$

dove $M = \max_x R^2(x)$ con $x \in X$. La funzione $\mu_A(\cdot)$ può essere definita come

$$\mu_A(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ \varphi(x) & \text{altrimenti.} \end{cases} \tag{3.10}$$

Alla funzione $\varphi(\cdot)$ è stato dato il nome di *fuzzificatore*, e può essere una qualsiasi funzione che decresce da uno a zero. Durante lo svolgimento di questo lavoro sono stati presi in considerazione i seguenti fuzzificatori, illustrati in Figura 3.1:

- Il *Fuzzificatore Crisp*, associa grado di apprtenenza nullo ai punti non appartenenti al core dell'insieme:

$$\mu_{\text{crisp}}(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ 0 & \text{altrimenti.} \end{cases} \tag{3.11}$$

- Il *Fuzzificatore Lineare*, decresce da uno a zero, linearmente, in base alla distanza $R^2(\cdot)$:

$$\mu_{\text{lin}}(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ \dfrac{M - R^2(x)}{M - R^2} & \text{se } R^2 < R^2(x) < M, \\ 0 & \text{altrimenti.} \end{cases} \tag{3.12}$$

- Il *Fuzzificatore Costante a Tratti*, costruisce una funzione di appartenenza a scalino i cui gradini corrispondono ai quattro quartili $(q_1, q_2, q_3, q_4,)$ della distribuzione distanze $R^2(\cdot)$ calcolati empiricamente:

$$\mu_{\text{qcost}}(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ \frac{3}{4} & \text{se } R^2 < R^2(x) < R^2 + q_1, \\ \frac{1}{2} & \text{se } R^2 < R^2(x) < R^2 + q_2, \\ \frac{1}{4} & \text{se } R^2 < R^2(x) < R^2 + q_3, \\ 0 & \text{altrimenti.} \end{cases} \tag{3.13}$$

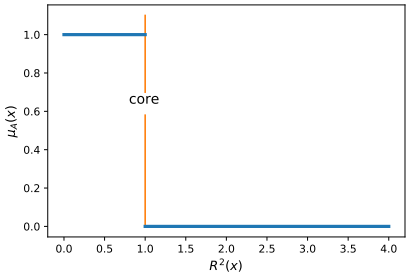
- Il *Fuzzificatore Lineare a Tratti*, corrisponde a una linearizzazione del fuzzificatore precedente:

$$\mu_{\text{qlin}}(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ -\dfrac{R^2(x) - R^2}{4q_1} + 1 & \text{se } R^2 < R^2(x) < R^2 + q_1, \\ -\dfrac{R^2(x) - R^2 - q_1}{4(q_2 - q_1)} + \dfrac{3}{4} & \text{se } R^2 < R^2(x) < R^2 + q_2, \\ -\dfrac{R^2(x) - R^2 - q_2}{4(q_3 - q_2)} + \dfrac{1}{2} & \text{se } R^2 < R^2(x) < R^2 + q_3, \\ -\dfrac{R^2(x) - R^2 - q_3}{4(q_4 - q_3)} + \dfrac{1}{4} & \text{se } R^2 < R^2(x) < R^2 + q_4, \\ 0 & \text{altrimenti.} \end{cases} \tag{3.14}$$

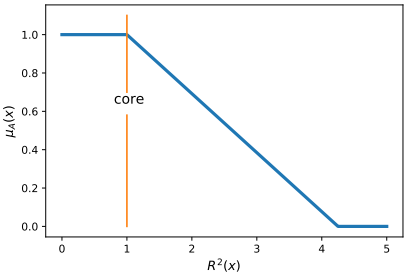
- Il *Fuzzificatore Esponenziale*, costruisce un decadimento esponenziale del grado di appartenenza, dipendente dal parametro $\alpha \in [0, 1]$:

$$\mu_{\text{exp}}(x) = \begin{cases} 1 & \text{se } R^2(x) \geq R^2, \\ \exp(\dfrac{\ln \alpha}{q_\alpha}(R^2(x) - R^2)) & \text{altrimenti.} \end{cases} \tag{3.15}$$

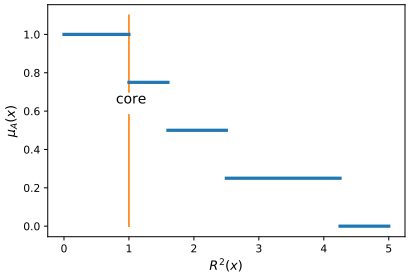
Il parametro α permette di legare il fuzzificatore esponenziale al corrispettivo quantile q della distribuzione delle distanze dal centro della sfera. In particolare, si avrà che il fuzzificatore vale esattamente α in corrispondenza di q. Informalmente, ciò significa che scegliendo un valore di α più vicino a 0 si ottiene un fuzzificatore più "schiacciato" sull'asse delle ascisse, mentre scegliendone uno più vicino a 1 si ha un fuzzificatore che tende meno velocemente a zero.



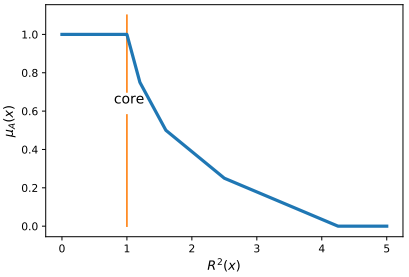
(a) Fuzzificazione crisp.



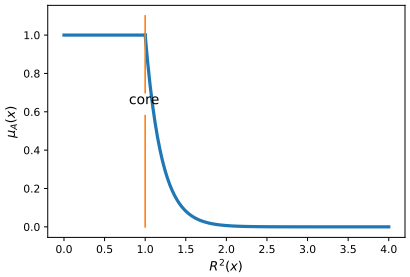
(b) Fuzzificazione lineare.



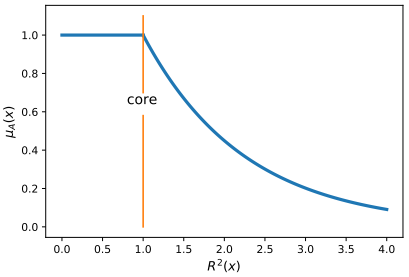
(c) Fuzzificazione costante a tratti, decrescente a ogni quartile.



(d) Fuzzificazione lineare a tratti, decrescente a ogni quartile.



(e) Fuzzificazione esponenziale con α vicino a zero.



(f) Fuzzificazione esponenziale con α vicino a uno.

Figura 3.1: Diversi esempi di fuzzificazione.

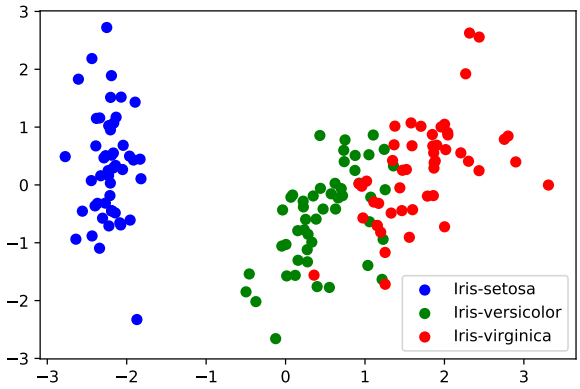


Figura 3.2: Dataset Iris proiettato nelle sue due componenti principali.

3.3 Costruire la funzione di appartenenza in assenza di membership: esempio con un benchmark

Il metodo è applicabile anche in mancanza dell'insieme U, cioè quando il grado di membership all'insieme da apprendere è sconosciuto per i punti appartenenti al training set. Una situazione del genere è verificabile in corrispondenza di un training set del tipo definito nel Paragrafo 2.1. Il dataset Iris [12], largamente usato nell'ambito dell'apprendimento automatico, consiste in 150 istanze del fiore Iris classificate in tre diverse specie: *setosa*, *virginica* e *versicolor*. In base a quattro differenti feature, le tre specie si posizionano nello spazio in modo che la classe setosa sia linearmente separabile dalle altre due che invece presentano una parziale sovrapposizione. La Figura 3.2 mostra il dataset nella sua rappresentazione bidimensionale dopo aver applicato la tecnica PCA [13] di riduzione della dimensionalità. Supponendo di voler apprendere per una delle classi, ad esempio setosa, il corrispondente insieme fuzzy, il metodo di apprendimento appena proposto può comunque essere utilizzato costruendo l'insieme U in modo tale che $\mu_i = 1$ se x_i appartiene alla classe setosa, $x_i = 0$ altrimenti. Lo stesso identico ragionamento può essere applicato per le restanti classi. Ne risultano così tre funzioni di appartenenza μ_{setosa}, $\mu_{virginica}$ e $\mu_{versicolor}$ apprese separatamente. Le Figure 3.3-3.5 mostrano il grafico di queste tre funzioni di appartenenza ottenute utilizzando un kernel gaussiano di parametro σ e fissando il parametro di tradeoff C. Si noti che, siccome le classi virginica e versicolor sono sovrapposte in parte e il criterio di fuzzificazione è basato sulla posizione dei punti nello spazio, i due insiemi fuzzy virginica e versicolor hanno un intersezione non vuota, come notato in Figura 3.4 e in Figura 3.5. Questo significa che per alcuni punti *di confine* x_c risulteranno $\mu_{virginica}(x_c) > 0$ e $\mu_{versicolor}(x_c) > 0$. Ciò non accade per l'insieme setosa che risulta disgiunto dagli altri due. Utilizzando come criterio di classifica-

zione di un punto il grado di appartenenza maggiore al corrispettivo insieme fuzzy ne risulta un'accuratezza di previsione su punti al di fuori del training set pari a circa il 95%.

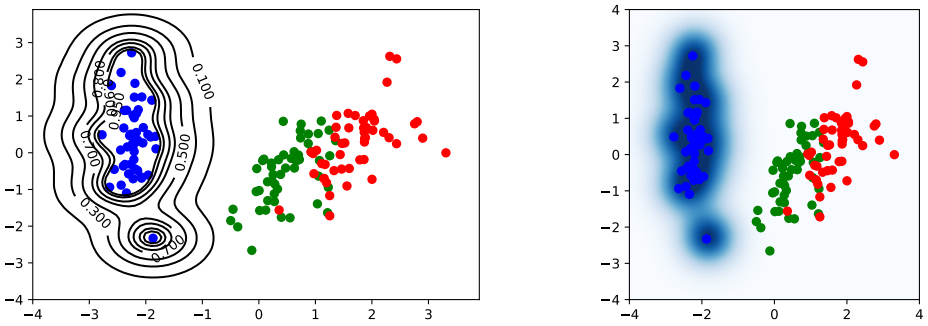


Figura 3.3: Contorni e mappa di calore dell'insieme fuzzy setosa. Durante l'apprendimento è stato usato il parametro $C = 10$ e un kernel gaussiano di parametro $\sigma = 0.5$.

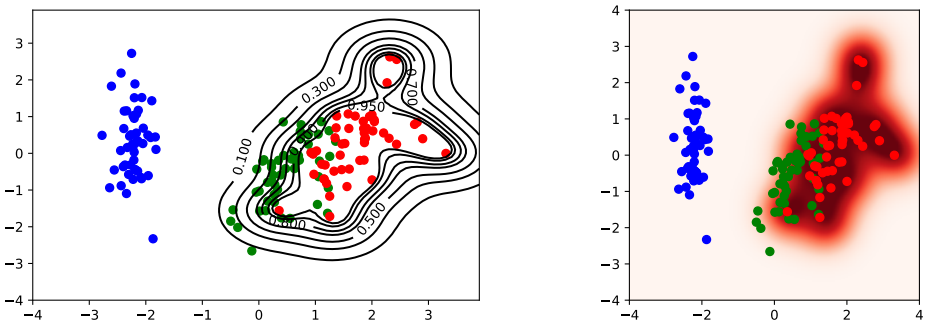


Figura 3.4: Contorni e mappa di calore dell'insieme fuzzy virginica. Durante l'apprendimento è stato usato il parametro $C = 10$ e un kernel gaussiano di parametro $\sigma = 0.5$.

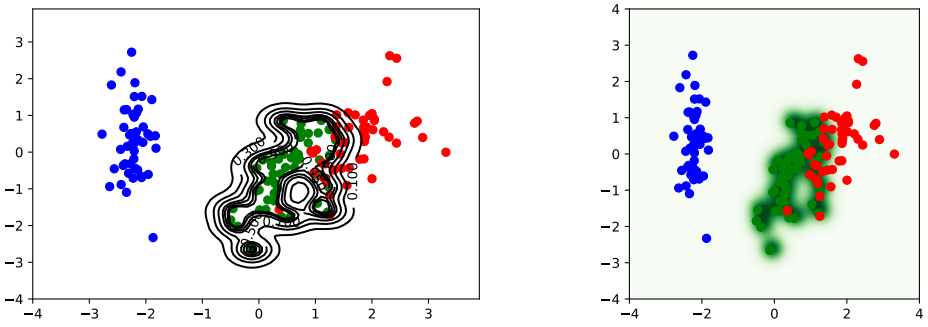


Figura 3.5: Contorni e mappa di calore dell'insieme fuzzy versicolor. Durante l'apprendimento è stato usato il parametro $C = 100$ e un kernel gaussiano di parametro $\sigma = 0.225$.

Capitolo 4

Apprendimento simultaneo di insiemi fuzzy

Il problema dell'apprendimento simultaneo di fuzzy set [14] consiste nella costruzione di un insieme $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ di m funzioni di appartenenza riguardanti altrettanti insiemi fuzzy $\{C_1, \cdots, C_m\}$. Dato un insieme di training $\{x_1, \cdots, x_n\}$ dove $x_i \in X \subseteq \Omega$, $\{C_1, \cdots, C_m\}$ è il risultato dell'algoritmo di support vector clustering applicato all'insieme X e le funzioni $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ sono il risultato dell'algoritmo di apprendimento illustrato nel Capitolo 4 applicato a ogni C_j. Il problema si compone di due livelli: individuare l'insieme $\{C_1, \cdots, C_m\}$ e costruire le funzioni $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$.

4.1 Individuazione degli insiemi fuzzy

Il primo passo nell'individuazione dei diversi insiemi fuzzy C_j è quello di costruire il problema di ottimizzazione alla base di 1-SVM per l'insieme X, che nella sua forma duale di Wolfe consiste nel minimizzare

$$\sum_{i=1}^{n} \beta_i K(x_i, x_i) - \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j K(x_i, x_j) \tag{4.1}$$

soggetta ai vincoli

$$\sum_{i=1}^{n} \beta_i = 1, \tag{4.2}$$

$$0 \leq \beta_i \leq C \; \forall \, i \in \{1, \cdots, n\}. \tag{4.3}$$

Il valore delle soluzioni $\{\beta_i^*, \cdots, \beta_n^*\}$, come visto nel Paragrafo 2.2, fornisce un metodo per l'individuazione dei vettori di supporto. Vale anche in questo caso che x_i è un vettore di supporto se $0 < \beta_i^* < C$. Una volta individuato l'insieme $S = \{x_i \mid 0 < \beta_i < C\}$ dei vettori di supporto, il quadrato del

raggio della sfera di inclusione risulta essere una media tra tutti i risultati della (2.31) calcolata per ogni vettore di supporto:

$$R^2 = \frac{1}{|S|} \sum_{x \in S} R^2(x). \tag{4.4}$$

La suddivisione in cluster viene effettuata tramite support vector clustering come descritto nel Paragrafo 2.3, ottenendo l'insieme $\{C_1, \cdots, C_m\}$ degli insiemi fuzzy. Per ogni insieme fuzzy trovato vale $C_j \subseteq X$. A questa prima procedura diamo il nome di *procedura di primo livello.*

4.2 Costruire le funzioni di appartenenza

Le singole funzioni di appartenenza sono ottenute applicando la metodologia descritta nel Capitolo 3. Per fare ciò, è necessario ricavare per ogni futuro insieme fuzzy il grado di appartenenza $\mu_{ij} = \mu_j(x_i)$ all'insieme C_j per ogni punto x dell'insieme di training, cioè il corrispettivo dell'insieme U nell'apprendimento di un unico fuzzy set (vedi Capitolo 3.). Nel caso di apprendimento simultaneo l'insieme U risulta essere più complesso: $U = \{(\mu_{11}, \cdots, \mu_{n1}), \cdots, (\mu_{1m}, \cdots, \mu_{nm})$. Per arrivare a U, a ogni elemento in $\{C_1, \cdots, C_m\}$ viene nuovamente applicato l'algoritmo di support vector clustering. Lo scopo dell'algoritmo in questo caso non è quello di suddividere l'insieme in cluster, ma quello di arrivare a conoscenza del raggio R_j^2 di ogni cluster C_j e di ogni funzione $R_j^2(\cdot)$. La generica funzione $R_j^2(x)$ calcola il quadrato della distanza dell'immagine del generico punto x dal centro del cluster C_j. Il raggio R_j^2 del cluster C_j questa volta non è più calcolato come una media tra le distanze delle immagini dei vettori di supporto dal centro, ma come il massimo tra queste:

$$R_j^2 = \max_{x \in S} [R_j^2(x)]. \tag{4.5}$$

La diversità nella formula del calcolo del raggio nei due diversi livelli della procedura dipende da una diversa necessità. Nel primo livello, utilizzare una media permette di avere un risultato più preciso sulla dimensione del raggio. Invece, utilizzare la media come punto di partenza per la fuzzificazione, porterebbe ad avere vettori di supporto, appartenenti al core del cluster, con un grado di appartenenza minore di uno, quando ai punti appartenenti a tale insieme è sempre assegnato il grado di appartenenza 1. Ottenute queste informazioni, il generico grado di appartenenza μ_{ij} è calcolato nel seguente modo:

$$\mu_{ij} = \begin{cases} 1 & \text{se } R_j^2(x_i) \leq R_j^2, \\ \varphi(R_j^2(x_i)) & \text{se } R_j^2(x_i) > R_j^2. \end{cases} \tag{4.6}$$

La funzione $\varphi(\cdot)$ è un fuzzificatore qualsiasi. A questo punto è possibile applicare a ogni cluster appartenente a $\{C_1, \cdots, C_m\}$ la procedura di apprendimento, ottenendo l'insieme desiderato di funzioni di appartenenza $\{\mu_1(\cdot), \cdots \mu_m(\cdot)\}$ ai diversi insiemi fuzzy $\{C_1, \cdots, C_m\}$. A questa seconda procedura diamo il nome di *procedura di secondo livello*.

4.3 Esperimenti

In questo capitolo viene presentato nel dettaglio l'algoritmo utilizzato per la sperimentazione della procedura di apprendimento simultaneo e i risultati ottenuti da tale sperimentazione.

4.3.1 Algoritmo di validazione

Al fine di testare la bontà dell'algoritmo di apprendimento simultaneo di insiemi fuzzy è stato costruito un sistema di validazione. In questo sistema viene iterata la ricerca di un modello ottimale tramite la prova di modelli differenti, come definito nell'Algoritmo 2. Per *modello* si intende un insieme di funzioni di appartenenza $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ apprese eseguendo la procedura di apprendimento simultaneo. Essendo ogni procedura dipendente dagli iperparametri C e σ indicheremo con $M_{C,\sigma}$ oppure $\{\mu_1(\cdot), \cdots \mu_m(\cdot)\}_{C,\sigma}$ il modello appreso tramite due specifici iperparametri. L'iperparametro C corrisponde alla variabile di tradeoff del problema 1-SVM, mentre l'iperparametro σ corrisponde al parametro del kernel gaussiano.
L'input del sistema di validazione è il seguente:

1. l'insieme X sul quale applicare la procedura,
2. l'insieme Y contenente le eventuali etichette assegnate ai punti contenuti in X,
3. un insieme di possibili parametri C,
4. un insieme di possibili parametri σ.

Inoltre è possibile selezionare alcuni parametri opzionali di esecuzione che, se non specificati, assumono un valore standard:

5. Il numero di ripetizioni I,
6. Un seme, *seed*, con cui inizializzare il generatore di permutazioni casuali.
7. Il numero delle componenti principali da considerarsi d.

8. La minima dimensione di un cluster accettabile *min*, come percentuale sul numero di oggetti presenti nel training set. I cluster che non raggiungono la dimensione minima non vengono trasformati in fuzzy set.

9. Il fuzzificatore $\varphi(\cdot)$ con cui assegnare i valori iniziali μ_{ij} di appartenenza in input al secondo livello.

10. Il fuzzificatore $\varphi'(\cdot)$ con cui attuare la fuzzificazione dell'insieme da apprendere.

11. Il fatto di forzare il numero di cluster a essere pari al numero di diversi valori per le etichette eventualmente presenti nel training set.

12. Il fatto di forzare tutte le eventuali etichette del training set a essere rappresentate da almeno un cluster.

13. Usare diversi parametri C e σ per i diversi livelli della procedura [1].

La divisione dell'insieme X in training set (80%), validation set (10%) e test set (10%), viene eseguita utilizzando una permutazione casuale dei dati, diversa per ogni iterazione. La scelta di un seme per l'inizializzazione del generatore di permutazioni casuali permette la ripetizione identica di un esperimento. Utilizzando la procedura di apprendimento simultaneo nella modalità supervisionata, a ogni funzione di appartenenza $\mu_j(\cdot)$ viene assegnata una tra le possibili etichette contenute in Y. Da qui in avanti si farà riferimento all'insieme $\{y_1, \cdots, y_m\}$, dove la generica etichetta y_j è quella assegnata alla funzione $\mu_j(\cdot)$ durante l'esecuzione supervisionata della procedura. Nello specifico, a ogni funzione di appartenenza $\mu_j(\cdot)$ viene assegnata l'etichetta più frequente tra quelle dei punti contenute nel corrispondente cluster C_j. A ogni punto del validation set vengono applicate le diverse funzioni $\mu_j(\cdot)$ e a ogni punto viene assegnata l'etichetta della funzione che restituisce il grado di appartenenza più alto. L'accuratezza a di ogni diverso modello $M_{C,\sigma}$ è calcolata come il rapporto tra il numero di etichette classificate correttamente e il numero di etichette totali presenti nel validation set:

$$a = \frac{\#\text{ etichette corrette}}{\#\text{ etichette totali}}. \tag{4.7}$$

Il modello con il valore di accuratezza più alto viene considerato il modello ottimale. In caso di pareggio tra più modelli ne viene scelto uno casualmente. Con gli iperparametri C_o e σ_o corrispondendi al modello ottimale M_{C_o,σ_o}

[1] L'utilizzo di valori diversi per gli iperparametri C e σ nei due differenti livelli della procedura produce modelli dipendenti non più da una coppia di iperparametri ma da una quadrupla $(C_0, \sigma_0, C_1, \sigma_1)$: la coppia (C_0, σ_0) utilizzata nel primo livello, la coppia (C_1, σ_1) utilizzata nel secondo. Tuttavia, i risultati ottenuti sperimentando quadruple di iperparametri non presentavano differenze sostanziali dai risultati ottenuti utilizzando coppie di iperparametri, producedo di fatto solamente un aumento del tempo impiegato per la ricerca del modello ottimale.

Algoritmo 2 Meccanismo di validazione

Proiettare l'insieme di dati nelle sue d componenti principali.
Costruire le tutte le possibili combinazioni dei parametri C e σ.
Inizializzare casualmente la variabile *seed*, se non definita.
i=0
while $i < I$ **do**
    Permutare casualmente l'insieme X.
    Suddividere X in training set, validation set e test set.
    best-p = []
    best-a = 0
    for ogni combinazione dei parametri C e σ **do**
        Apprendere simultaneamente $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ dal training set.
        Assegnare a ogni $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ una tra le possibili etichette.
        Calcolare l'accuratezza a del modello $\{\mu_1, \cdots, \mu_m\}$ utilizzando il validation set.
        if $a >$ *best-a* **then**
            best-p $= (C, \sigma)$.
            best-a $= a$
        end if
    end for
    Con *best-p* apprendere $\{\mu_1, \cdots, \mu_m\}$ dal training set unito con il validation set.
    Assegnare a ogni $\{\mu_1, \cdots, \mu_m\}$ un'etichetta y_j.
    Calcolare l'accuratezza a_o del modello ottimale $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ utilizzando il test set.
    i=i+1
end while

viene rieseguito l'apprendimento delle $\{\mu_1(\cdot), \cdots, \mu_m(\cdot)\}$ utilizzando il training set sommato al validation set e successivamente calcolata l'accuratezza a_o sul test set.
Quando si sceglie di non forzare il numero di cluster al numero di etichette differenti contenute in Y, è possibile che il numero m dei cluster ottenuti dal primo livello di clusterizzazione sia maggiore del numero di etichette differenti. Ne consegue che esisterà almeno una coppia di insiemi fuzzy a cui è stata assegnata la stessa etichetta. Il metodo di assegnazione di un punto del validation (test) set a un insieme fuzzy è basato sul grado di appartenenza più alto. In virtù della (1.8), apprendendo quindi i diversi insiemi fuzzy con questa modalità, il generico cluster C_l rappresentante della classe etichettata l risulta essere

$$C_l = \bigcup_{y_i=l} C_i. \tag{4.8}$$

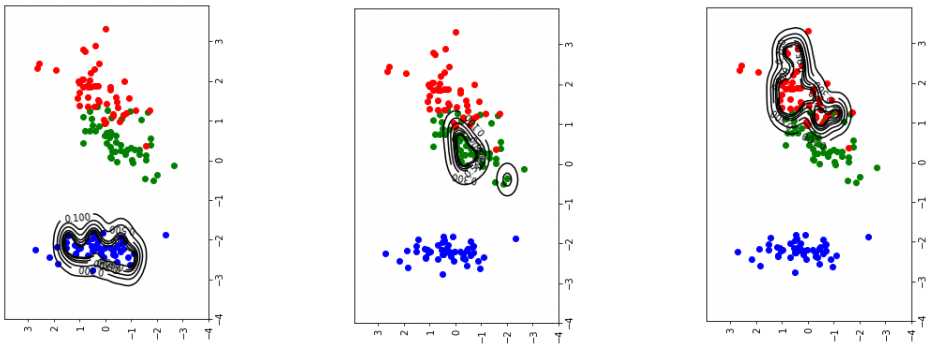
Quando al contrario si sceglie di forzare l'uguaglianza tra il numero di cluster e il numero di etichette differenti, vengono mantenuti i primi k cluster per grandezza e si scartano gli altri, che non vengono trasformati in insiemi fuzzy. Qualora il numero m di cluster individuati è tale che $m < k$, il modello non viene considerato nella ricerca di quello ottimale. Analogamente, scegliere di forzare le differenti etichette a essere rappresentate da almeno un insieme fuzzy fa in modo che i modelli tali per cui esiste almeno un'etichetta $l \in Y$ non presente nell'insieme $\{y_1, \cdots, y_m\}$ non vengano considerati.
Va specificato che per implementare la soluzione del problema di ottimizzazione definito da (4.1) è stata utilizzata la libreria Gurobi Optimizer [15]. Il risultato di questa ottimizzazione è un vettore $\{\beta_1^*, \cdots, \beta_n^*\}$ i cui elementi non sono mai uguali né a zero né a C, sebbene il numero di vettori di supporto sia strettamente minore del numero di dati a disposizione. Al fine di ottenere una soluzione consona è stata definita una soglia $\varepsilon = 10^{-4}$ tale che se $0 < \beta_i < 0 + \varepsilon$, allora β_i viene approssimato a zero. Analogamente se $C - \varepsilon < \beta_i < C$, allora β_i viene approssimato a C. I vettori di supporto risultanti dal problema di minimizzazione risultano quindi essere quegli x_i la cui corrispondente β_i è tale che $0 + \varepsilon < \beta_i < C - \varepsilon$.

4.3.2 Risultati

Di seguito vengono mostrati i risultati sperimentali della procedura di apprendimento simultaneo descritta nei paragrafi precedenti. Nei primi esperimenti sono stati analizzati iperparametri in un insieme delle potenze di 10: Da 10^{-2} a 10^2. Successivamente, in base ai risultati ottenuti, il range è stato raffinato. In ogni esperimento è stato utilizzato il dataset Iris mostrato nel Capitolo 3. unitamente a un numero di ripetizioni $I = 10$. Nelle Tabelle 4.1-4.5, che riassumono i risultati ottenuti, la colonna Dim. indica il numero di componenti principali utilizzate durante l'esperimento. Le colonne Avg.

Tabella 4.1: Risultato di 10 iterazioni della procedura di apprendimento simultaneo (SVC) a confronto con l'algoritmo fuzzy c-means (FCM).

	SVF				FCM			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.82	0.05	0.8	0.11	0.83	0.01	0.80	0.05
3	0.87	0.03	0.83	0.07	0.83	0.02	0.84	0.07
4	0.85	0.05	0.89	0.09	0.83	0.01	0.84	0.08



(a) setosa (b) versicolor (c) virginica

Figura 4.1: Contorno degli insiemi fuzzy setosa, versicolor e virginica. Durante questo esperimento sono stati usati gradidi membership binari e un fuzzificatore lineare.

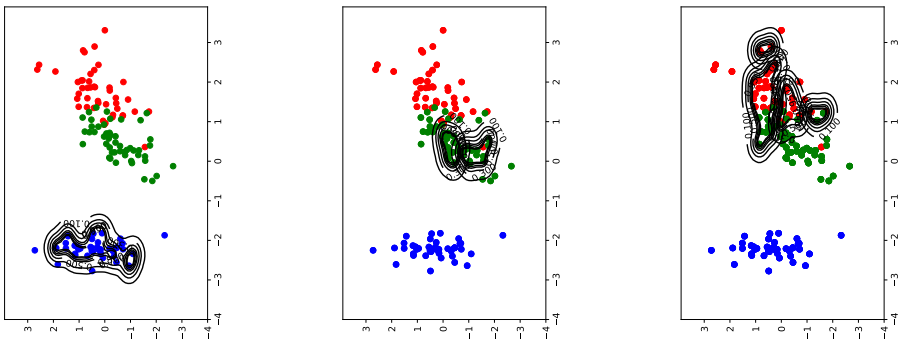
e StDev. indicano rispetitvamente la media e la deviazione standard dell'accuratezza calcolata sull'insieme di training e sull'insieme di test.

La Tabella 4.1 contiene i risultati sperimentali del confronto della procedura di apprendimento simultaneo tramite support vector clustering con l'algoritmo *fuzzy c-means* [16]. Questo algoritmo divide il dataset in un numero prefissato c di insiemi fuzzy. Per questo esperimento l'algoritmo di apprendimento simultaneo è stato quindi forzato a ottenere $c = 3$ cluster, uno per ogni diversa etichetta presente nel dataset Iris. In Figura 4.1 è mostrato il contorno, rappresentato come curve di livello sul grado di membership, degli insiemi fuzzy ottenuti secondo questi criteri dal metodo di apprendimento simultaneo. I risultati in Tabella 4.1 sono leggermente a favore dell'algoritmo proposto.

Nella Tabella 4.2 è mostrato il confronto tra l'utilizzo di membership binarie e membership linearmente decrescenti durante il clustering di secondo livello. I risultati ottenuti sono a favore delle prime. Come precedentemente notato, in presenza di $\mu_i = 0$ (3.2) tende a escludere dal core dell'insieme fuzzy il punto x_i. Ciò non avviene in presenza di $\mu_i > 0$. In presenza di clu-

Tabella 4.2: Risultato di 10 iterazioni della procedura di apprendimento simultaneo utilizzando μ di tipo binario e lineare. .

	μ binarie				μ lineari			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.83	0.80	0.84	0.17	0.80	0.04	0.80	0.08
3	0.90	0.06	0.88	0.06	0.84	0.07	0.84	0.15
4	0.84	0.10	0.81	0.13	0.77	0.20	0.67	0.24



(a) setosa (b) versicolor (c) virginica

Figura 4.2: Contorno dei tre insiemi fuzzy setosa, versicolor e virginica appresi mediante valori di membership μ di tipo binario. Gli insiemi versicolor (4.2(b)) e virginica (4.2(c)) sono individuati dall'unione di più insiemi fuzzy.

ster sovrapposti, l'utilizzo di gradi di appartenenza μ di tipo binario separa più facilmente i punti appartenenti a classi differenti, rendendo migliore il risultato della successiva classificazione. Va notato come l'accuratezza media calcolata nelle quattro dimensioni con μ lineari risulti sproporzionata rispetto alle altre e presenti una più alta deviazione standard. Non forzando il numero di cluster a essere uguale a quello delle etichette, da un lato si ottengono insiemi fuzzy più precisi, definiti come l'unione di insiemi più piccoli (Figure 4.2(b)-4.2(c)), dall'altro è possibile incorrere in modelli composti da un numero di insiemi fuzzy minori del numero delle diverse etichette. La conseguenza di questo evento è un calo nella precisione di classificazione. Questo spiega il valore medio fuori scala e la più alta deviazione standard presenti nell'ultima riga della Tabella 4.2. La Figura 4.2 mostra un esempio dei contorni degli insiemi fuzzy ottenuti durante questa sperimentazione.

Le Tabelle 4.3 e 4.4 contengono i risultati ottenuti variando il metodo di fuzzificazione tra quelli precedentemente introdotti. I risultati peggiori si hanno in corrispondenza delle fuzzificazioni crisp e costante a tratti. Queste funzioni infatti presentano dei confini ben precisi di passaggio da un grado di membership a un altro e mal si adattano alla rappresentazione di un in-

sieme fuzzy. Diversamente, le funzioni che decrescono in maniera continua ottengono buoni risultati. In Figura 4.3 sono visualizzati i contorni degli insiemi fuzzy setosa, versicolor e virginica appresi mediante una fuzzificazione di tipo crisp. Da notare come, in questo caso, i confini degli insiemi risultino più evidenti che nelle altre figure proposte. In Figura 4.4 è possibile vedere la mappa di calore degli stessi insiemi fuzzy, appresi con fuzzificazione costante a tratti. Le diverse tonalità di colore che demarcano il diminuire del grado di appartenenza, anche in questo caso, si differenziano nettamente.

Tabella 4.3: Risultato di 10 iterazioni della procedura di apprendimento simultaneo utilizzando diversi fuzzificatori.

	Crisp				QuantCost.			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.58	0.14	0.42	0.15	0.71	0.05	0.61	0.13
3	0.48	0.07	0.38	0.14	0.49	0.11	0.44	0.16
4	0.48	0.07	0.34	0.05	0.41	0.05	0.33	0.15
	Lineare				QuantLin.			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.83	0.80	0.84	0.17	0.76	0.10	0.80	0.15
3	0.90	0.06	0.88	0.06	0.83	0.17	0.80	0.16
4	0.84	0.10	0.81	0.13	0.70	0.14	0.70	0.20

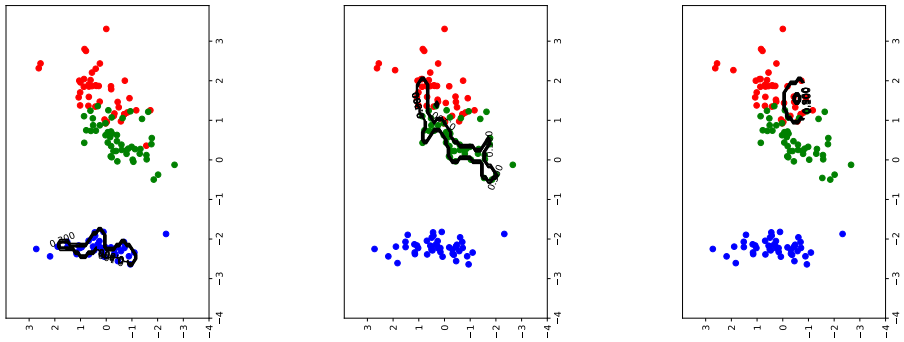
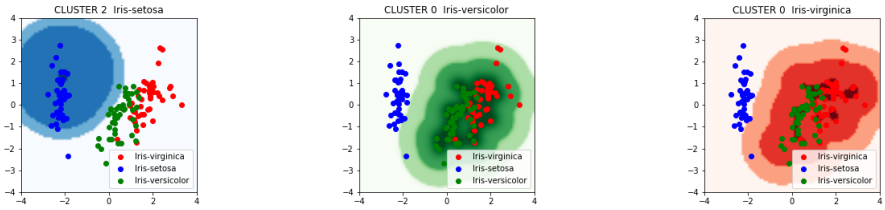


Figura 4.3: Contorno dei tre insiemi fuzzy setosa, versicolor e virginica appresi mediante una fuzzificazione di tipo crisp.



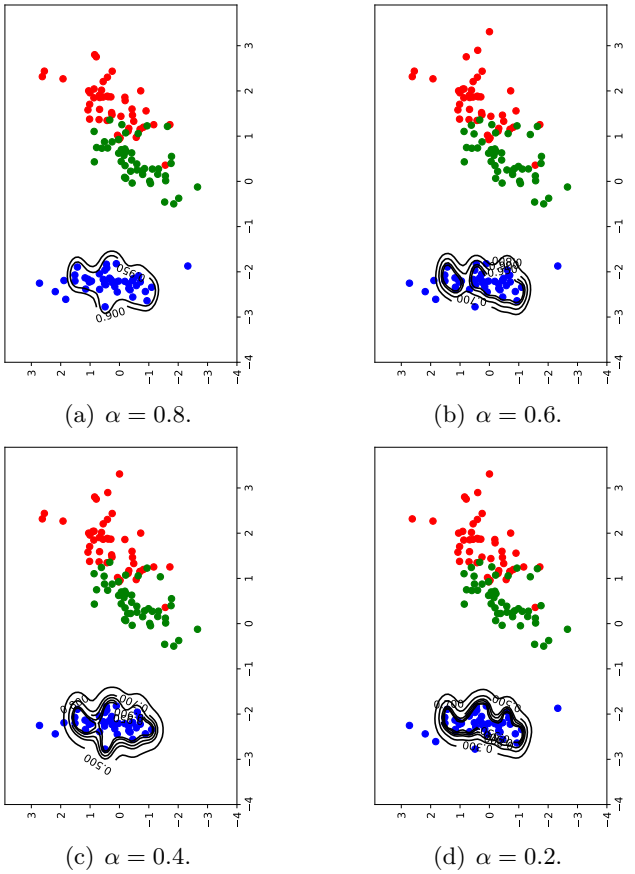
(a) setosa (b) versicolor (c) virginica

Figura 4.4: Mappa di calore dei tre insiemi fuzzy setosa, versicolor e virginica con l'utilizzo di una fuzzificazione costante a tratti.

Come esposto in precedenza, valori di α prossimi allo zero fanno sì che la funzione del fuzzificatore esponenziale risulti schiacciata sull'asse delle ascisse, mentre con valori più vicini a uno la velocità con cui la funzione tende allo zero diminuisce. Sostanzialmente, al diminuire di α il fuzzificatore esponenziale tende sempre di più ad assumere la forma del fuzzificatore crisp e, allo stesso modo, i contorni degli insiemi fuzzy ottenuti dal processo di apprendimento tendono a farsi sempre più netti. Questo fenomeno è mostrato in Figura 4.5, dove sono messi a confronto i contorni dell'insieme setosa appresi mediante diversi esperimenti in cui è stato variato il parametro α del fuzzificatore esponenziale. In Figura 4.5(a), in corrispondenza di $\alpha = 0.8$, il grado di appartenenza dei punti appena al di fuori del core è pari a 0.9. Questo valore diminuisce per $\alpha = 0.6$ a 0.7 (Figura 4.5(b)) e succesivamente a 0.5 per $\alpha = 0.4$ (Figura 4.5(b)). Infine, in Figura 4.5(d) sono mostrati i contorni dell'insieme setosa appreso con un fuzzificatore esponenziale di parametro $\alpha = 0.2$, dove oltre il confine del core dell'insieme fuzzy il grado di appartenenza risulta essere pari a 0.3, segno del continuo delineamento dei contorni dell'insieme.

Tabella 4.4: Risultato di 10 iterazioni della procedura di apprendimento simultaneo utilizzando il fuzzificatore esponenziale con un diverso parametro α .

	$\alpha = 0.2$				$\alpha = 0.4.$			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.82	0.08	0.75	0.15	0.80	0.12	0.80	0.16
3	0.80	0.18	0.75	0.19	0.82	0.10	0.77	0.12
4	0.63	0.0.09	0.56	0.13	0.55	0.19	0.44	0.14
	$\alpha = 0.6$				$\alpha = 0.8.$			
	Train		Test		Train		Test	
Dim.	Avg.	StDev.	Media	StDev.	Avg.	StDev.	Media	StDev.
2	0.86	0.05	0.80	0.12	0.77	0.19	0.68	0. 30
3	0.71	0.31	0.76	0.21	0.84	0.10	0.82	0.13
4	0.63	0.10	0.60	0.14	0.60	0.14	0.52	0.19



(a) $\alpha = 0.8$. (b) $\alpha = 0.6$.

(c) $\alpha = 0.4$. (d) $\alpha = 0.2$.

Figura 4.5: Contorno dell'insieme fuzzy setosa ottenuto mediante fuzzificatori esponenziali di diverso parametro α.

Infine, in Tabella 4.5 sono presentati i risultati ottenuti applicando l'algoritmo di clustering *possibilistic c-means* [17] in sostituzione a 1-SVM. Nel caso possibilistico i gradi di appartenenza non dipendono dalla distanza ma sono un'interpretazione probabilistica e possibilistica.

Tabella 4.5: Risultato di 10 iterazioni della procedura di apprendimento simultaneo utilizzando il metodo di clustering possibilistico.

	Train		Test	
Dim.	Avg.	StDev.	Avg.	StDev.
2	0.86	0.03	0.84	0.12
3	0.83	0.05	0.80	0.15
4	0.82	0.01	0.84	0.07

4.4 Complessità

Prendendo come riferimento l'algoritmo SMO [18] per la risoluzione di problemi di ottimizzazione quadratici, la complessità temporale per la risoluzione del problema di ottimizzazione alla base del support vector clustering risulta, anche per l'apprendimento simultaneo di insiemi fuzzy, pari a $O(n^2)$, dove n è il numero di esempi presenti nel training set. Indicando rispettivamente con n_{sv} il numero di support vector individuati a valle del problema di ottimizzazione e con n_{out} il numero di outlier, la complessità della fase di ricerca dei cluster, dovendo costruire e controllare le condizioni di appartenenza allo stesso cluster per ogni possibile coppia di punti, risulta essere $O((n - n_{out})^2)$. È possibile ridurre la complessità della fase di clustering controllando i segmenti congiungenti le sole coppie di punti contenenti un vettore di supporto senza perdite nell'accuratezza [10]. Tale complessità risulta quindi diminuire a $O((n - n_{out})n_{sv})$. Il valore n_{out} è controllato dal parametro C come osservato in [10] mediante la disequazione

$$n_{out} < \frac{1}{nC}. \tag{4.9}$$

È necessario evidenziare che questo procedimento viene applicato ulteriormente, per ogni cluster, in quello che è stato chiamato secondo livello dell'apprendimento simultaneo di fuzzy set.
Si noti infine che un metodo recentemente proposto, basandosi su proprietà topologiche delle funzioni kernel [19], riduce la complessità della clusterizzazione a $O(n)$ e, in specifiche situazioni, a $O(nlog(n))$. L'accuratezza della classificazione ne risulta però deteriorata.

Conclusioni

Questa tesi ha proposto l'estensione di un metodo di apprendimento per funzioni di appartenenza a un insieme fuzzy. In particolare, è stato adattato un algoritmo proposto in letteratura (basato sul concetto di support vector clustering) al fine di apprendere in modo simultaneo le funzioni di appartenenza di molteplici insiemi fuzzy definiti in uno stesso spazio. Una volta definita la procedura in maniera analitica, è stato costruito un ambiente per la sperimentazione e l'analisi dell'algoritmo in modo supervisionato, al fine di testare la procedura secondo diverse opzioni. In particolare, per la fase di fuzzificazione, che risulta essere centrale nell'apprendimento di insiemi fuzzy, sono state utilizzate differenti tipologie di funzioni. I risultati sperimentali hanno mostrato come l'utilizzo di funzioni più vicine al concetto classico di insieme mal si adattano alla costruzione di insiemi sovrapposti, come per il caso delle classi *versicolor* e *virginica* nel dataset Iris, i cui confini si mischiano diventando così difficilmente delineabili. Per questo tipo di insiemi, l'alternativa fuzzy risulta essere la più accurata. Inoltre, in facoltà dei risultati ottenuti dal confronto con l'algoritmo *fuzzy c-means* si può affermare che il metodo proposto sia una valida alternativa nell'inferenza di insiemi fuzzy. Il processo di fuzzificazione risulta essere indipendente dal metodo di clustering utilizzato per l'individuazione dei diversi insiemi fuzzy. Questo apre molteplici possibilità di sviluppo, per esempio utilizzando metodi migliorati per il support vector clustering oppure algoritmi di tipo possibilistico: i primi nell'ordine di diminuire la complessità computazionale della procedura, mentre i secondi in virtù dei primi risultati sperimentali ottenuti, che superano in media quelli ottenuti tramite support vector clustering. Infine, un ulteriore sviluppo del lavoro proposto potrebbe essere quello di estendere l'algoritmo al fine di apprendere fuzzy set di tipo 2.

Bibliografia

[1] Lotfi A Zadeh et al. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[2] Jerry M Mendel and RI Bob John. Type-2 fuzzy sets made simple. *IEEE Transactions on fuzzy systems*, 10(2):117–127, 2002.

[3] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] Roger Fletcher. Practical methods of optimization john wiley & sons. *New York*, 80, 1987.

[6] HW Kuhn and AW Tucker. Proceedings of 2nd berkeley symposium, 1951.

[7] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[8] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70, 2004.

[9] Philip Wolfe. A duality theorem for non-linear programming. *Quarterly of applied mathematics*, 19(3):239–244, 1961.

[10] Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[11] Dario Malchiodi and Witold Pedrycz. Learning membership functions for fuzzy sets through modified support vector clustering. In *International Workshop on Fuzzy Logic and Applications*, pages 52–59. Springer, 2013.

[12] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[13] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[14] Luca Cermenati, Dario Malchiodi, and Anna Maria Zanaboni. Simultaneous learning of fuzzy sets. *Proceedings of the WIRN 2018 conference*, 2018. in stampa.

[15] Gurobi Optimization. Inc.,"gurobi optimizer reference manual," 2015. *URL: http://www. gurobi. com*, 2014.

[16] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[17] Maurizio Filippone, Francesco Masulli, and Stefano Rovetta. Applying the possibilistic c-means algorithm in kernel-induced spaces. *IEEE Transactions on Fuzzy Systems*, 18(3):572–584, 2010.

[18] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[19] Jaewook Lee and Daewon Lee. An improved cluster labeling method for support vector clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 27(3):461–464, 2005.