Diversi approcci al problema della classificazione

Federico Poloni <f.poloni@sns.it>

22 maggio 2006

1 Il problema della classificazione

1.1 Introduzione

In diversi contesti applicativi è interessante riuscire a costruire un sistema che "impari" a classificare correttamente una serie di campioni, deducendo quali sono i parametri più significativi che li caratterizzano. Ad esempio:

- in campo medico, un sistema che, ottenuti un numero sufficiente di dati clinici, sia in grado di scoprire se un paziente è affetto da una certa patologia, o quale sia la gravità della stessa: ad esempio, in [1] ci si propone di determinare in base ai risultati quantitativi di una biopsia se un caso di tumore è benigno o maligno.

- un "classificatore di testi" che sia in grado di dire se un testo è attinente o no a un argomento prestabilito in base solo ai termini che vi compaiono.

- un "classificatore di immagini" che, dati alcuni punti di un semplice disegno (per esempio una figura geometrica o una scacchiera), sia in grado di ricostruire il *pattern* sottostante, come in alcuni esempi di [2].

In particolare il primo problema ha fornito negli ultimi tempi una notevole spinta a questo settore della matematica.

Il paradigma utilizzato per riuscire a costruire queste *learning machines* è quello di utilizzare un insieme di "dati di allenamento" (*training set*) già correttamente classificati e di costruire basandosi su di essi una "funzione di classificazione" che poi, si spera, sia in grado di classificare appropriatamente anche casi al di fuori di questo insieme.

Possiamo formalizzare così il problema della classificazione in presenza di due alternative: sia X un generico insieme (solitamente $X = \mathbb{R}^n$); abbiamo un insieme di coppie

$$T = \{(x_i, y_i)\,,\, x = 1, 2, \ldots, m\} \subset X \times \{-1, +1\},$$

detto *training set*, e vogliamo trovare una funzione $f : X \to \{-1, 1\}$ all'interno di una classe $\mathcal{F}$ il cui grafico passi per le coppie date, o si avvicini ad esse il più possibile.

Una volta trovata questa funzione, possiamo applicarla per classificare nuovi punti al di fuori del *training set*: a seconda che $f(x)$ valga $+1$ o -1, classificheremo il punto $x \in X$ in una delle due classi.

Il caso più semplice è quello della *classificazione lineare*, cioè il problema di determinare un iperpiano $\{x \mid \langle w, x\rangle + b = 0\}$ in $X = \mathbb{R}^n$ in modo che un insieme di punti dati P stia nel semispazio positivo delimitato da esso e un altro insieme di punti N stia nel semispazio negativo: in questo caso allora $f = \operatorname{sgn}(\langle w, x\rangle + b)$.

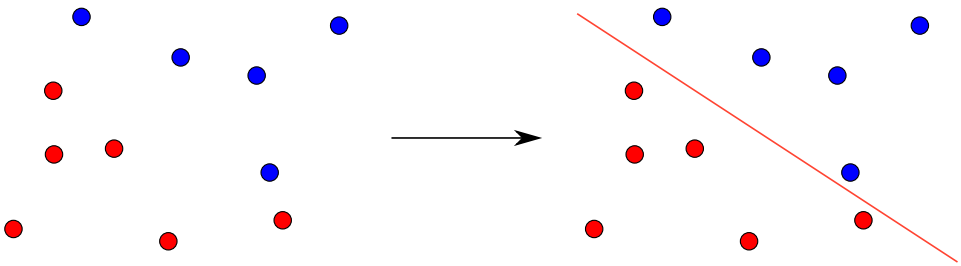

Figura 1: Un problema di classificazione lineare: trovare il piano che separa i punti rossi da quelli blu

Spesso, nel caso di dati presi dal mondo reale, una classificazione esatta è impossibile, in quanto l'errore presente intrinsecamente nei dati impedisce di trovare un piano che separi esattamente i punti "positivi" da quelli "negativi" (figura 2). In questo caso si possono adottare diverse strategie per trovare l'approssimazione più soddisfacente: per esempio, si può tentare di minimizzare il numero di errori, o la distanza complessiva dall'iperpiano dei punti classificati erroneamente. È stato dimostrato che in generale il problema di minimizzare il *numero* di errori di classificazione è NP-completo, quindi è pressoché impossibile trovare algoritmi efficienti per trattarlo. Parte del problema diventa allora quello di stabilire un'adeguata "funzione di errore" da minimizzare che consenta di raggiungere risultati soddisfacenti con una complessità accettabile.

1.2 Dalla classificazione lineare ai kernel

Per una classificazione più efficace, spesso è necessario adottare metodi di separazione diversi da quelli lineari: per esempio, vorremmo poter trovare una superficie sferica, o il grafico di un polinomio cubico, che separino i punti dei due insiemi. Questo corrisponde a variare l'insieme $\mathcal{F}$ delle funzioni ammissibili di cui parlavamo nella sezione 1.1: parlare di *classificazione lineare* (cioè tramite iperpiani) equivale ad affermare che le funzioni ammis-

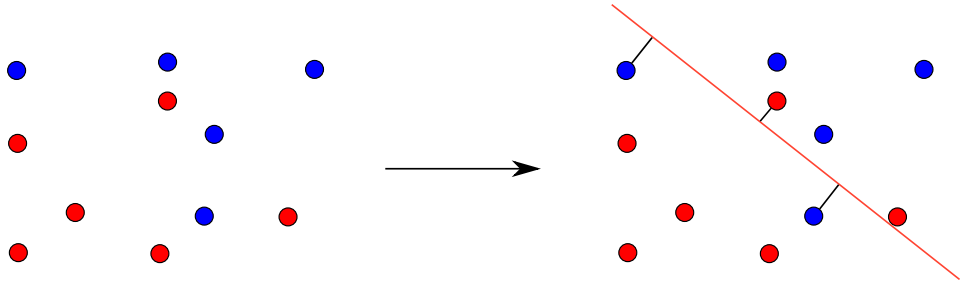

Figura 2: Un problema di classificazione lineare con errore: le linee nere rappresentano la distanza dal piano dei punti mal classificati. Non esiste un piano che classifichi correttamente tutti i punti dati.

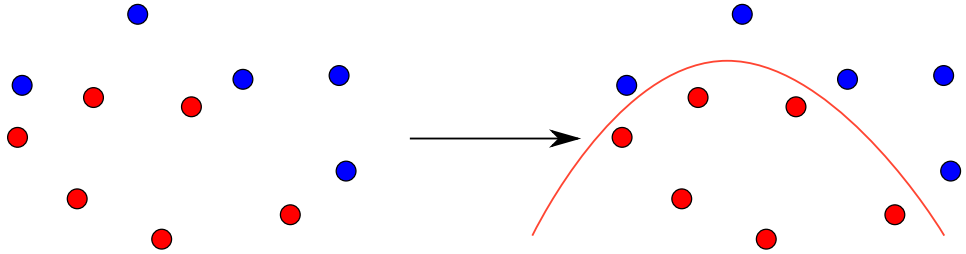

Figura 3: Con una funzione di grado superiore, siamo in grado di separare meglio i punti rispetto ad un iperpiano

sibili sono quelle della forma $x \mapsto \operatorname{sgn}(\langle w, x\rangle + b)$, con $w \in \mathbb{R}^n, b \in \mathbb{R}$, che identificano i punti che stanno da una parte o dall'altra di un iperpiano.

Potremmo per esempio voler utilizzare come separatori non solo gli iperpiani, ma le superfici polinomiali di grado $\leq k$, ad esempio per $k = 2$ tutte le funzioni della forma $x \mapsto \operatorname{sgn}(\langle x, Ax\rangle + \langle w, x\rangle + b)$. Questo ci fornisce una classificazione più accurata, a costo però di una maggiore complessità di calcolo.

Un modo efficace di ottenere i benefici di questa classificazione continuando però a utilizzare gli algoritmi del caso lineare è la tecnica dello spazio immagine (*feature space*), che illustriamo ora partendo da un esempio. Supponiamo di avere un problema di classificazione in cui $X = \mathbb{R}^2$; a tutti i punti da classificare possiamo applicare questa trasformazione:

$$\phi : \mathbb{R}^2 \to \mathbb{R}^5 : \quad (x, y) \mapsto (x^2, xy, y^2, x, y)$$

e poi utilizzare un algoritmo di classificazione lineare per trovare un iperpiano in $\mathbb{R}^5$ che "separi bene" le immagini dei punti dati. Per come è costruita la mappa ϕ, un iperpiano nell'immagine equivale a una funzione del tipo $ax^2 + bxy + cy^2 + dx + ey + f$, cioè a un generico polinomio di secondo grado nelle coordinate dei punti dello spazio di partenza; e non è difficile provare che in effetti si tratta del polinomio di secondo grado che meglio separa i

punti dati. Quindi, al solo costo di aumentare la dimensione dello spazio, siamo riusciti a ricondurre un problema di classificazione quadratica a uno lineare.

Nello stesso modo possiamo mappare uno spazio di partenza X in un opportuno *spazio immagine* Y di dimensione maggiore attraverso $\phi : X \to Y$ e cercare un iperpiano separatore in Y. A seconda della scelta di Y, questo equivale a trovare funzioni di classificazione in classi diverse, più ampie della classe delle funzioni lineari su X.

Questo approccio rivela la sua potenza quando è combinato con le versioni *duali* di alcuni degli algoritmi che presenteremo: tali versioni infatti utilizzano i punti x_i solo attraverso prodotti scalari del tipo $\langle x_i, x_j \rangle$. Quindi non dobbiamo calcolare esplicitamente $\phi(x_i)$ e $\phi(x_j)$ ma soltanto $\langle \phi(x_i), \phi(x_j) \rangle$, cosa che possiamo fare attraverso una *funzione kernel*

$$K_\phi : X \times X \to \mathbb{R} : \quad x_i, x_j \mapsto K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

In questo modo, possiamo utilizzare direttamente spazi immagine di dimensione molto grande senza dover mai lavorare direttamente con le loro coordinate, ma utilizzando solo la funzione kernel. Se scegliamo tale funzione accuratamente, possiamo far sì che il calcolo di K sia notevolmente più veloce del calcolo esplicito di $\phi(x_i)$ e $\phi(x_j)$. Un caso in cui questo avviene è quello del kernel polinomiale

$$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} : \quad x, y \mapsto (\langle x, y \rangle + 1)^k$$

che fornisce un prodotto scalare nello spazio immagine formato da tutti i monomi di grado $\leq k$ nelle n coordinate dei punti di partenza. Per esempio, per $n = k = 2$ abbiamo

$$K\left((x_1, y_1), (x_2, y_2)\right) = (x_1 x_2 + y_1 y_2 + 1)^2$$

che, come è semplice verificare, corrisponde al prodotto scalare euclideo nello spazio immagine determinato da

$$\phi(x, y) = \begin{pmatrix} x^2 \\ y^2 \\ \sqrt{2}xy \\ \sqrt{2}x \\ \sqrt{2}y \\ 1 \end{pmatrix}$$

e quindi fornisce una classificazione mediante una superficie quadratica, come nell'esempio fatto poco sopra.

Un altro kernel utilizzato frequentemente nelle applicazioni è il *kernel gaussiano*, dato da $K(x, y) = \exp(-\|x - y\|^2 / \sigma^2)$.

1.3 Più di due scelte

Nei problemi che abbiamo considerato finora la funzione incognita aveva immagine in $\{-1, +1\}$, cioè i punti da classificare potevano appartenere a due sole categorie, per esempio nelle applicazioni mediche potevamo distinguere i pazienti sani da quelli malati. Spesso però siamo interessati a un'analisi più raffinata, per esempio vorremmo prevedere da quale malattia è affetto il paziente, o se si tratta di una forma lieve o grave della stessa. Vorremmo pertanto essere in grado di estendere i nostri algoritmi di classificazione binaria a lavorare su più insiemi.

Formalizzando, abbiamo un numero finito di insiemi di *training points* già classificati nelle classi $P_1, P_2, \ldots, P_k \subset X$ e siamo in cerca di una funzione $f : X \to \{1, 2, \ldots, k\}$ all'interno di una classe opportuna $\mathcal{F}$ che li classifichi correttamente, cioè tale che per tutti gli $i = 1, 2, \ldots, k$ valga $f(x) = i \; \forall x \in P_i$.

Un approccio frequente in letteratura [3] è quello di risolvere m problemi binari, separando di volta in volta l'insieme P_i dal suo complementare $\bigcup_{j \neq i} P_j$. Se la classificazione è esatta e gli insiemi sono "ben separabili", su ogni possibile ingresso $x \in X$ solo una delle classificazioni darà risultato positivo, dicendoci quindi in quale dei P_i va classificato il punto. In caso contrario, potremmo ottenere come responso l'appartenenza a due o più diversi P_i, o anche a nessuno di essi: a seconda delle applicazioni, potremmo voler riportare il caso come "dubbio" perché venga sottoposto ad un'ulteriore analisi, oppure sviluppare un'euristica per prendere una decisione in questi casi.

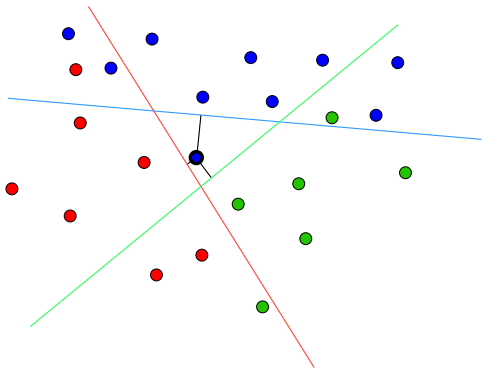

Figura 4: Un esempio di classificazione multipla con errori: il punto evidenziato non sta in nessuno dei tre semispazi associati ai P_i, ma con l'euristica presentata verrebbe (erroneamente) classificato assieme ai punti rossi, perché l'iperpiano relativo è quello a lui più vicino

Un buon esempio di euristica è la seguente: noi otteniamo per ognuno degli m sottoproblemi binari un iperpiano $\langle w_i x \rangle + b_i$ che dovrebbe individuare

i punti di P_i secondo questa regola:

$$\langle w_i x \rangle + b_i > 0 \Rightarrow x \in P_i \tag{1}$$
$$\langle w_i x \rangle + b_i < 0 \Rightarrow x \in P_j, \quad j \neq i \tag{2}$$

Se i coefficienti di questi iperpiani sono normalizzati in qualche modo (ad esempio $\|w_i\| = 1$), allora i valori della funzione $w_i x + b_i$ sono confrontabili al variare di i; nel caso in cui si scelga di porre $\|w_i\|_2 = 1$, essi indicano proprio la distanza euclidea tra il punto x e l'iperpiano relativo. Quindi una strategia buona è quella di classificare x basandoci sul *massimo* di tali valori:

$$f(x) := \text{argmax}_{i=1,2,\ldots,m}(w_i x + b_i),$$

cioè scegliere l'insieme P_i per cui il vincolo corrispondente è soddisfatto con il maggior margine.

2 Algoritmi per la classificazione lineare binaria

In questa sezione presenteremo alcune classi di algoritmi per la classificazione *lineare binaria*. L'enunciato del problema e la notazione sono quindi quelli della sezione 1.1: sia $X = \mathbb{R}^n$, $T = \{x_i \mid i = 1, \ldots, m\} \subset X$ il *training set* e $y_i \in \{0, 1\}$ per $i = 1, \ldots, m$; si richiede di trovare una coppia (w, b) tale che $y_i = \text{sgn}(\langle w, x_i \rangle + b)$, o, quando questo non è possibile, minimizzando una opportuna funzione di errore che stabiliremo di volta in volta.

2.1 Il perceptron

Storicamente, il primo algoritmo iterativo per la classificazione lineare fu quello proposto da F. Rosenblatt negli anni '50, noto con il nome di *perceptron* [4]. L'idea che ne sta alla base è quella di partire da un vettore di pesi $(w, b) = 0$ e aggiornarlo man mano ogni volta che l'iperpiano $wx + b = 0$ sbaglia a classificare uno dei punti di T.

2.1.1 Versione primale

Per enunciare l'algoritmo, innanzitutto introduciamo una semplificazione formale applicando una omogeneizzazione sullo spazio di partenza $X = \mathbb{R}^n$: ad ogni vettore x giustapponiamo una componente costante di valore[1] $R \neq 0$ in modo da ottenere $\hat{x} = (x \mid R)'$: in questo modo l'equazione dell'iperpiano separante diventa

$$\langle w, x \rangle + \frac{b}{R} R = \langle \hat{w}, \hat{x} \rangle ,$$

[1] Solitamente si sceglie $R = \max_{i=1\ldots m} \|x_i\|$; argomenti di probabilità in [2] suggeriscono che si tratti della scelta che fornisce la classificazione più "affidabile" partendo da esempi uniformemente distribuiti in X.

dove $\hat{w} = \left(w \mid \frac{b}{R}\right)'$.

Con la notazione introdotta, il perceptron assume questa forma:

```
ŵ = 0; k=0;
repeat
  mistakes←0;
  for each x_i ∈ T
    if y_i ⟨ŵ, x̂_i⟩ ≤ 0
      mistakes←mistakes+1;
      ŵ←ŵ + η y_i x̂_i;
    end if
  end for
until mistakes=0
```

dove $\eta > 0$ è un parametro detto *learning rate*.

Si noti che la quantità $y_i \langle \hat{w}, \hat{x_i} \rangle$, detta *margine* della coppia (x_i, y_i), è negativa (o nulla) quando l'iperpiano determinato da $\hat{w}$ sbaglia a classificare il valore di *training* (x_i, y_i): quindi quando si verifica un errore l'algoritmo modifica il vettore $\hat{w}$ in modo che il margine di x_i aumenti della quantità $\eta \langle x_i, x_i \rangle$ e quindi si avvicini ad ogni iterazione a diventare positivo.

Non è difficile dimostrare che se esiste un iperpiano in grado di classificare correttamente i punti dati (cioè se essi sono *separabili linearmente*), l'algoritmo finisce e fornisce una soluzione al problema. In generale però questo non accade quando i punti non sono classificabili esattamente (come nell'esempio in figura 2); questo rende il *perceptron* inaccettabile nella maggior parte delle applicazioni. Tuttavia esso condivide, almeno a un livello elementare, alcune delle proprietà delle *support vector machines* che costituiscono uno strumento molto più avanzato per risolvere il problema della classificazione.

2.1.2 Versione duale

Il vettore $\hat{w}$ utilizzato nell'algoritmo del *perceptron*, per come viene costruito lungo l'algoritmo, in ogni momento è una combinazione lineare dei vettori di input $\hat{x}_i$:

$$\hat{w} = \sum_{i=1}^{m} \alpha_i \hat{x}_i$$

dove ogni α_i è uguale al numero di volte che il vettore x_i è stato classificato erroneamente durante l'algoritmo moltiplicato per ηy_i.

Il *perceptron* può anche essere formulato utilizzando questi α_i in luogo del vettore w_i:

```
α = 0; k=0;
repeat
  mistakes←0;
```

```
  for each x_i ∈ T
    if y_i Σ_{j=1}^{m} α_j ⟨x̂_j, x̂_i⟩ ≤ 0
      mistakes←mistakes+1;
      α_i←α_i + η y_i;
    end if
  end for
until mistakes=0
```

Questa formulazione presenta diversi vantaggi:

- Essa è un analogo della rappresentazione in coordinate duali dei problemi di programmazione lineare. Come accade nella programmazione lineare, anche qui le variabili α_i portano con sé delle informazioni aggiuntive rilevanti nel contesto del problema: in questo caso, esse sono maggiori (in valore assoluto) quanto più è stato difficile classificare il punto dato, quindi individuano immediatamente i punti più "significativi" o "al limite" all'interno dell'insieme di training.

- In questa forma, i vettori di x_i entrano in gioco solo attraverso i prodotti scalari $\langle x_j, x_i \rangle$: quindi, come osservavamo nella sezione 1.2, l'algoritmo si applica in modo veloce e naturale al caso di problemi linearizzati mediante una funzione kernel. In più, la funzione di classificazione fornita dall'algoritmo

  $$f(x) = \operatorname{sgn} \sum_{i=j}^{m} \alpha_j \langle x_j, x \rangle$$

  è nella stessa forma, quindi si può utilizzare per classificare nuovi punti $x \in X \setminus T$ senza doverne calcolare esplicitamente le coordinate nello spazio immagine.

- Nella pratica, molti degli α_i sono nulli, quindi non solo nella sommatoria scritta si può evitare il calcolo di molti dei prodotti scalari, ma otteniamo naturalmente una caratterizzazione dei soli punti "utili" ai fini della classificazione.

2.2 Programmazione lineare e *support vector machines*

La strategia che esaminiamo ora è quella di ridurre il problema della classificazione a un problema di programmazione lineare o quadratica. Poiché questi problemi sono già stati ampiamente studiati sia dal punto di vista teorico che da quello applicativo degli algoritmi, possiamo beneficiare immediatamente dei risultati ottenuti nella teoria dell'ottimizzazione. In particolare, questo approccio produce algoritmi computazionalmente efficienti e adattabili a un ampio numero di casi in cui non i dati non sono classificabili senza errore.

I metodi più utilizzati all'interno di questa classe sono quelli detti *support vector machines* (SVM).

2.2.1 Il *maximal margin classifier*

Il modello più semplice di *support vector machine* è il *maximal margin classifier*, che si ottiene minimizzando la distanza dei punti dall'iperpiano classificante. Formalmente, definiamo il *margine geometrico* di un punto (x_i, y_i) rispetto a un iperpiano (w, b) come la distanza in norma-2 del punto dall'iperpiano classificante, presa con segno positivo se il punto è classificato correttamente e negativo se è classificato erroneamente. Utilizzando la formula per la distanza di un punto da un piano, si ha

$$m(x_i, y_i) = \frac{y_i(\langle w, x_i \rangle + b)}{\|w\|_2}.$$

Il *maximal margin classifier* si ottiene cercando l'iperpiano che massimizza il minimo dei margini al variare dei punti $(x_i, y_i) \in T$:

$$\max_{w,b} \min_{(x_i,y_i)\in T} m(x_i, y_i).$$

Vediamo ora come con una manipolazione algebrica questo problema si può ridurre in un problema di programmazione quadratica: supponiamo che i punti siano separabili, quindi che esistano iperpiani per cui tutti i margini sono positivi; dato un iperpiano siffatto, normalizziamolo moltiplicando w e b per una costante in modo che

$$\min_{(x_i,y_i)\in T} y_i(\langle w, x_i \rangle + b) = 1.$$

Ora, per la scelta appena fatta il minimo margine della distribuzione è $\frac{1}{\|w\|_2}$; quindi massimizzare il margine equivale a risolvere questo problema di programmazione quadratica:

$$\begin{cases} \min_{w,b} \langle w, w \rangle, \\ y_i(\langle w, x_i \rangle + b) \geq 1 \quad i = 1, \ldots, m. \end{cases}$$

Tra le proprietà di questo classificatore citiamo le seguenti:

- La regione ammissibile del problema è non vuota se e solo se i punti di T sono separabili; quindi l'algoritmo si può applicare in tutti e soli i casi in cui i punti sono separabili, come accadeva con il perceptron.
- Come per il perceptron, anche questo problema ammette una formulazione duale in cui intervengono solo i prodotti scalari $\langle x_i, x_j \rangle$ e la funzione risultante è del tipo

$$f(x) = \sum_{j=1}^{m} \alpha_j \langle x_j, x \rangle,$$

cioè adatta per lavorare con una funzione kernel.

- Dalle condizioni di Karush-Kuhn-Tucker segue che i punti per cui $\alpha_j > 0$ sono un sottoinsieme dei punti per cui $y_i(\langle w, x_i\rangle + b) = 1$, cioè i punti per cui si raggiunge il minimo della distanza dall'iperpiano classificatore. Nel caso tipico, questi punti sono solo una piccola quantità e quindi il calcolo della funzione f è abbastanza agevole. Tali punti sono detti *support vectors*; si noti che il risultato della classificazione dipende solo da questi punti, quindi anche rimuovendo tutti gli altri elementi di T si otterrebbe come risultato lo stesso iperpiano.

2.2.2 Il *soft margin classifier*

La maggiore limitazione dei due algoritmi finora esposti è quella di funzionare correttamente solo nel caso in cui i punti sono separabili da un iperpiano. Nel caso di problemi reali, questa limitazione è catastrofica, in quanto basta un singolo errore sperimentale o un dato discordante per rendere impossibile la classificazione. Pertanto si rende necessario sviluppare un metodo di classificazione più robusto, che sia in grado di fornire un iperpiano classificatore sufficientemente buono anche nel caso di dati non perfettamente separabili.

Nel contesto delle *support vector machines*, questo si ottiene applicando un trucco classico della programmazione lineare, vale a dire l'introduzione di variabili di scarto: modifichiamo i vincoli del problema in modo da consentire che alcuni punti non soddisfino le condizioni di separazione, fallendo per una quantità ξ_i:

$$\begin{aligned} &y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \\ &xi_i \geq 0. \end{aligned}$$

Se consentiamo valori arbitrariamente alti per gli ξ_i, è semplice vedere che si possono ottenere valori piccoli a piacere per $\|w\|$; quindi dobbiamo introdurre una limitazione anche sugli xi_i; è naturale considerare a questo proposito le due seguenti funzioni obbiettivo:

$$\min_{w,b} \left(\langle w, w\rangle + C \sum_{i=1}^{m} \|\xi_i\|_2 \right), \tag{3}$$

$$\min_{w,b} \left(\langle w, w\rangle + C \sum_{i=1}^{m} |\xi_i| \right). \tag{4}$$

con C costante positiva arbitraria. Di fatto, nelle applicazioni si usa provare diversi valori di C e poi testare a posteriori quale di questi fornisca l'approssimazione più ragionevole a seconda del problema specifico.

I problemi di minimo diventano quindi

$$\begin{cases} \min_{w,b} \langle w, w\rangle + C \|\xi\| \\ y_i(\langle w, x_i\rangle + b) \geq 1 - \xi \quad i = 1 \dots m. \end{cases}$$

e a seconda della scelta della norma-2 o della norma-1 per il vettore ξ otteniamo l'una o l'altra delle due formulazioni (3) e (4), dette rispettivamente *2-norm soft margin classifier* e *1-norm soft margin classifier*.

Entrambe le funzioni danno origine a dei problemi di minimo la cui soluzione è assicurata anche nel caso di problemi di classificazione con un *training set* non separabile e che, se espressi in forma duale, hanno le stesse buone proprietà del *maximal margin*. In particolare, il secondo dà origine a una funzione obiettivo $f(x) = \sum_{j=1}^{m} \alpha_j \langle x_j, x \rangle$ in cui vale la limitazione $\alpha_i \leq C$, che si può pensare come un limite massimo al peso di ogni punto di training nel classificatore; questa scelta limita quindi naturalmente l'influenza di punti che corrispondono a errori sperimentali e che altrimenti potrebbero modificare notevolmente la funzione obbiettivo.

2.2.3 Altri tipi di separazione

In [5] si tenta di utilizzare al posto di un iperpiano una superficie sferica che separi i due insiemi di punti. Fissando il centro della sfera e facendone variare solo il raggio, si ottiene un problema particolarmente trattabile la cui soluzione richiede soltanto $O(m \log m)$ operazioni, dove $m = |T|$. La complessità degli altri metodi di programmazione quadratica qui esposti, in generale maggiore, dipende dall'algoritmo utilizzato per la soluzione del QP e in generale non è semplice da descrivere. Questa scelta ha però lo svantaggio di fornire una classificazione molto meno flessibile: avere fissato a priori il centro della sfera implica conoscere approssimativamente qual è l'unico dato rilevante ai fini della classificazione, cioè la distanza dei punti di T dal centro della sfera, e quindi annulla molti dei benefici delle SVM come "macchine in grado di imparare".

2.3 Problemi agli autovalori

Un altro possibile approccio alla classificazione (si veda ad esempio [6]) è quello di cercare non un iperpiano separatore tra i vari esempi ma un iperpiano al quale i punti di uno dei due insiemi da separare (ma non quelli dell'altro) si avvicinino il più possibile: in formule, se chiamiamo $A = \{x \mid (x, 1) \in T\}$ e $B = \{x \mid (x, -1) \in T\}$ i due insiemi da separare, quello che cerchiamo è

$$\min_{w,b} \frac{\sum_{x \in A} \|\langle w, x \rangle + b\|^2}{\sum_{x \in B} \|\langle w, x \rangle + b\|^2}$$

(la scelta della norma-2 al quadrato è necessaria per le manipolazioni che intendiamo fare). Se eliminiamo i termini costanti b con la tecnica utilizzata nella sezione 2.1.1 e indichiamo con A e B le matrici che hanno come righe rispettivamente i punti di A e B (espressi come vettori di $\mathbb{R}^n$), il problema

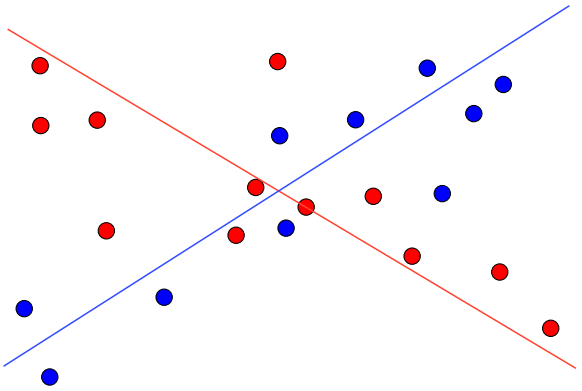

Figura 5: Un modello di classificazione in cui cerchiamo non un iperpiano separatore, ma i due iperpiani che meglio approssimano la distribuzione dei punti forniti: la retta disegnata in rosso cerca di descrivere la posizione dei punti rossi, quella in blu la posizione dei punti blu.

assume questa forma:

$$\min_{\hat{w}\in\mathbb{R}^{n+1}} \frac{\hat{w}'A'A\hat{w}}{\hat{w}'B'B\hat{w}}$$

(qui x' indica il trasposto del vettore o della matrice x). Indicando $G = A'A$ e $H = B'B$, abbiamo che la quantità da minimizzare è quello che viene detto *quoziente di Rayleigh* del problema agli autovalori generalizzato $Gx = \lambda Hx$.

Ricordiamo che il quoziente di Rayleigh di un problema agli autovalori generalizzato $Gx = \lambda Hx$ con G e H hermitiane è la quantità $r(G, H, x)\frac{x'Gx}{x'hx}$; esprimendola in funzione degli autovalori si ottiene che essa è sempre compresa tra $|\lambda_{min}|$ e $|\lambda_{max}|$ (con λ_{min} e λ_{max} si indicano l'autovettore di modulo minimo e quello di modulo massimo del problema), e assume tali valori quando x è parallelo rispettivamente a x_{min} e a x_{max} (gli autovettori relativi a λ_{min} e λ_{max}).

Quindi abbiamo che il vettore $\hat{w}$ che minimizza la (2.3) è proprio x_{min}, che corrisponde all'iperpiano che meglio identifica i punti di A rispetto a quelli di B nel senso indicato sopra; analogamente, $\hat{w} = x_{max}$ fornisce l'iperpiano che meglio identifica B rispetto ad A.

Abbiamo allora ricondotto il problema di classificazione a un problema agli autovalori generalizzato, problema ampiamente studiato nel contesto dell'algebra lineare numerica e per cui sono noti diversi algoritmi risolutivi (metodo QZ, metodo di Lanczos). La soluzione numerica del problema può incontrare difficoltà in alcuni casi degeneri (per esempio, quando c'è un vettore x tale che $Gx = Hx = 0$); in tal caso si possono adottare tecniche di algebra lineare numerica per migliorare la stabilità dell'algoritmo. In ogni caso, poiché il calcolo di autovalori e autovettori è possibile solo in modo approssimato, le questioni di approssimazione e le tecniche di

shifting tipiche dell'algebra lineare numerica rivestono un ruolo importante nell'implementazione dell'algoritmo.

Si noti che tali iperpiani sono ortogonali (in quanto corrispondono ad autovettori di autovalore distinto), quindi si tratta di una classificazione di tipo completamente diverso rispetto a quella fornita dalle SVM. Quella che otteniamo in questo caso è una funzione che ci dice quanto un punto è "vicino" alle caratteristiche dell'insieme A o all'insieme B; in casi particolari (x vicino all'intersezione tra i due iperpiani classificanti) può succedere che *entrambe* queste funzioni abbiano un valore sufficientemente piccolo. A seconda del contesto applicativo, rilevare questa somiglianza può avere un ruolo importante nella classificazione.

3 Conclusioni

Gli algoritmi che abbiamo presentato forniscono due tecniche risolutive per il problema della classificazione le cui radici si trovano in diverse branche della matematica applicata. Implementare e studiare entrambe le tecniche diventa importante per riuscire a trattare al meglio problemi di classificazione sempre più complessi e di dimensioni sempre maggiori.

Il beneficio derivante dalla risoluzione di questi problemi applicativi con tecniche matematiche è grande, in quanto ci consente di applicare metodi matematici anche in campi in cui un'analisi esatta del problema è impossibile e parrebbe necessario affidarsi a tecniche empiriche. L'applicazione alla catalogazione di testi appare promettente, specialmente ora che le problematiche relative alla ricerca su internet sono un importante soggetto di ricerca; sono estremamente interessanti anche le applicazioni in campo biomedico, in cui l'utilizzo della matematica non è ancora ampiamente diffuso e può portare enormi miglioramenti nell'analisi dei dati rispetto alle tecniche empiriche usate storicamente.

Riferimenti bibliografici

[1] O. L. Mangasarian, W. N. Street, W. H. Wolberg, *Breast cancer diagnosis and prognosis via linear programming.* Technical report, 1994.

[2] N. Cristianini, J. Shawe-Taylor, *An Introduction to support vector machines.* Cambridge University press, 2000.

[3] F. Giannessi, *Teoremi di separazione e problemi di classificazione.* Università di Pisa, 2005.

[4] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review 65, 1959.

[5] A. Astorino, M. Gaudioso, *Spherical separation and kernel transformations for classifications problems*. Rivista e anno sconosciuti.

[6] M. R. Guarracino, *On classification methods for mathematical models of learning*. Quaderni dell'università di Pisa, 2005.