Metodologia di boosting in ambito di problemi multiclasse

Candidato: Giacomo D'Angelo
Relatore: Prof. Fabrizio Malfanti
Correlatore: Prof.ssa Eva Riccomagno

Statistica Matematica e trattamento Informatico dei Dati
Università degli studi di Genova
A.A. 2013/2014

Introduzione

L'argomento trattato nella mia tesi riguarda Adaboost nella sua versione multiclasse, avendo avuto un primo approccio nella mia esperienza di tirocinio.

Adaboost

- Adaboost, diminutivo di Adaptive Boosting, consiste nel considerare classificatori deboli (weak), per combinarli e ottenerne uno più performante.
- È molto accurato, semplice e con diverse applicazioni possibili, risulta meno suscettibile al rischio di overfitting rispetto ad altri algoritmi di apprendimento.
- I difetti che presenta questo filtro includono il fatto che risulta sensibile agli outliers e ai “noisy” data, ovvero ai dati sporchi.

Adaboost

- Sia $D = (x_1, y_1), (x_2, y_2), .., (x_m, y_m)$ con $Y = \{-1, +1\}$.
- Si inizializza il vettore dei pesi: $D_1(i) = 1/m$
- Il processo viene effettuato per T iterazioni
- Sia $\mathcal{L}$ weak learner iniziale una funzione di due variabili (D, D_t) a valori in Y

Adaboost

1. $h_t = \mathcal{L}(D, D_t)$ viene fatto il train di un weak learner h_t da D usando la distribuzione D_t
2. $\varepsilon_t = Pr_i \sim D_i[h_t(x_i \neq y_i)]$ misura l'errore di h_t
3. $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$ determina il peso di h_t
4. $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} exp(-\alpha_t) & h_t(x_i) = y_i \\ exp(\alpha_t) & h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
   Aggiorna la distribuzione, dove Z_t è un fattore di normalizzazione che permette a D_{t+1} di essere una distribuzione.

L'output sarà:

$$h_{fin}(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

Adaboost - Considerazioni

- L'algoritmo Adaboost viene applicato per casi dicotomici, ovvero dove le possibili uscite (i possibili risultati) possono essere due (es. vero/falso, sì/no, 1/0, ecc.).
- Esistono problemi in cui gli stati di classificazione sono più di due, nasce quindi l'esigenza di adattare il filtro per supportare queste situazioni.

Adaboost

- Nel corso dell'esperienza di tirocinio svolta presso Intelligrate srl, si ha affrontato un problema inerente alla predizioni di risultati calcistici.
- Calcio, problema multi-classe: i possibili eventi di qualsiasi partita sono tre (vittoria squadra A, pareggio, vittoria squadra B).
- Vengono analizzate le principali varianti dell'algoritmo Adaboost nel caso multivariato.

Adaboost.M1

È la versione più semplice tra gli adaboost multiclasse.

- Per una sequenza di m esempi di training $(x_1, y_1), ...(x_m, y_m)$ con etichette $y_i \in Y = \{1, ..., k\}$
- Si inizializza il vettore dei pesi: $D_1(i) = 1/m$
- Il processo viene effettuato per t=1,...,T
- Sia $\mathcal{L}$ weak learner iniziale

Adaboost.M1

1. $h_t = \mathcal{L}(D, D_t)$ viene fatto il train di un weak learner h_t da D usando la distribuzione D_t
2. $\varepsilon = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ misura l'errore di h_t
   Se ε >1/2, allora T=t-1 e il programma abortisce
3. $\alpha_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ determina il peso di h_t
4. $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \alpha_t & h_t(x_i) = y_i \\ 1 & altrimenti \end{cases}$
   Aggiorna la distribuzione, dove Z_t è un fattore di normalizzazione (scelto in modo tale che D_{t+1} sia una distribuzione).

L'output sarà:

$$h_{fin}(x) = \underset{y\in Y}{\operatorname{argmax}} \sum_{t:h_t(x)=y} log \frac{1}{\alpha_t}$$

Adaboost.M1-Considerazioni

Il principale svantaggio dell'Adaboost.M1 è quello di essere incapace di trattare le weak ipotesi con errore superiore a $1/2$. L'errore previsto dall'ipotesi che genera casualmente l'etichetta è 1-$1/$k.

- $k = 2$: ipotesi weak devono essere migliori della scelta casuale
- $k > 2$: la richiesta che l'errore sia minore di $1/2$ è piuttosto forte e spesso difficile da conoscere.

Adaboost.M2

- Viene permesso al weak learner di generare più ipotesi dove, piuttosto che identificare una singola etichetta in Y, sceglie un set di "plausibili" etichette.
- Si permette inoltre al weak learner di dare un grado di plausibilità, così ogni ipotesi weak darà un vettore $[0, 1]^k$.
- Mentre l'algoritmo weak acquista maggior potenza espressiva, si pone un requisito sulle performance delle ipotesi weak piu' complesso.
- Nasce una misura di errore più sofisticata: la pseudo-loss, calcolata sull'insieme di coppie di tutti gli esempi e etichette sbagliate.

Adaboost.M2

Una *mislabel* è una coppia (*i,y*) dove *i* è l'indice dell'esempio di training e *y* è un'etichetta sbagliata associata all'esempio *i*. Sia *B* l'insieme di tutte le *mislabel* $B = \{(i, y) : i \in \{1, ..., m\}, y \neq y_i\}$. Una distribuzione mislabel è una distribuzione definita su *B*.

- Per una sequenza di *m* esempi $(x_1, y_1), ...(x_m, y_m)$ con etichette $y_i \in Y = \{1, ..., k\}$ si inizializza il vettore dei pesi: $D_1(i) = 1/|B| per (i, y) \in B$
- Il processo è svolto per T volte specificando il numero di iterazioni
- Sia $\mathcal{L}$ weak learner iniziale

Adaboost.M2

1. Viene chiamato il weak learner $\mathcal{L}$, fornendolo di una distribuzione D_t
2. Torna indietro un'ipotesi $h_t : X \times Y \rightarrow [0, 1]$
3. Viene calcolata la pseudo-loss di h_t:
$$\varepsilon = \frac{1}{2} \sum_{(i,y)\in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$
4. $\alpha_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ determina il peso di h_t
5. Si aggiorna D_t:
$$D_{t+1}(i, y) = \frac{D_t(i,y)}{Z_t} \alpha_t^{(1/2)(h_t(x_i,y_i)-h_t(x_i,y))}$$
dove Z_t è un fattore di normalizzazione che rende D_{t+1} una distribuzione.

L'output sarà:

$$h_{fin}(x) = \underset{y\in Y}{\text{argmax}} \sum_{t=1}^{T} log \frac{1}{\alpha_t} h_t(x, y)$$

Concetto Single/Multi-label

- La classificazione del singolo evento solitamente rimane sempre la stessa (single label).

Squadra A	Squadra B	Esito
Juventus	Chievo	1
Juventus	Chievo	1
Juventus	Chievo	X

- L'oggetto partita Juventus-Chievo, nella fase di addestramento, ogni volta che sarà giocata, avrà la possibilità di essere classificata in maniera diversa, e quindi potrà assumere più di un tipo di label (multi-label).

Forward Stagewise Additive Modeling

Molti modelli di classificazione e regressione possono essere scritti come combinazione lineare di modelli più semplici:

$$f(x) = \sum_{m=1}^{M} \beta_m b_m(x, \gamma_m)$$

Dove:

- x è un dato di input
- $\{\beta_m, \gamma_m\}$ sono parametri del modello
- $b_m(x, \gamma_m)$ sono altre funzioni arbitrarie di x.

Forward Stagewise Additive Modeling

- Generalmente, $\{\beta_m, \gamma_m\}$ sono stimate per minimizzare alcune loss function L.
- Spesso questo procedimento risulta essere piuttosto complicato, se però si ottimizza la funzione su una funzione base del tipo:
  $$min \sum_{i=1}^{M} L(y_i, \beta b_m(x_i, \gamma)$$
  il problema può essere risolto facilmente.
- Aggiungere sequenzialmente nuove funzioni base per l'espansione della funzione $f(x)$ senza cambiare i parametri che sono stati aggiunti.

Forward Stagewise Additive Modeling

L'adaboost fitta un modello addittivo, attraverso il modello forward stagewise, dove:

- La funzione base b_m è un classificatore binario
- La funzione oggetto è la loss esponenziale

$$L(y, f) = e^{-yf(x)}$$

Il prossimo algoritmo si basa su queste considerazioni.

Adaboost SAMME

- Si inizializza il vettore dei pesi: $D_1(i) = 1/m$
- Il processo viene effettuato per T iterazioni
- Sia $\mathcal{L}$ il classificatore weak iniziale

1. Viene chiamato il weak learner $\mathcal{L}$
2. In risposta, esso genera h_t per i dati di training usando i pesi
3. $\varepsilon_t = \sum_{i=1}^{m} D_t(i)\mathbb{1}(c_i \neq h_t(x_i))/\sum_{i=1}^{m} D_t(i)$ misura l'errore
4. $$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t} + log(K-1) \tag{1}$$

Adaboost SAMME

- Si assegna:

$$D_t(i) \leftarrow D_t(i)\,\exp(\alpha_t \mathbb{1}(y_i \neq h_t(x_i))),\ \mathrm{i} = 1,\ \ldots,\ \mathrm{m}$$

- Ri-normalizza $D_i(t)$

L'output sarà:

$$C(x) = \operatorname*{argmax}_k \sum_{(t=1)}^{T} \alpha_t \mathbb{1}(h_t(x) = k)$$

Adaboost SAMME

- SAMME assomiglia molto alle versioni Adaboost, con la principale differenza in (1).
- In modo tale da avere α_t positivo, si ha necessità solo che $1 - \varepsilon_t >$1/K, o che l'accuratezza di ogni classificatore weak sia migliore della classificazione casuale.
- Il termine additivo $log(K - 1)$ non è artificiale e crea un nuovo algoritmo equivalente a fittare un modello additivo in forward stagewise attraverso la loss function esponenziale.

Giustificazione teorica

- Si costruisce una nuova funzione di perdita esponenziale, per adattarla al caso multi-classe.
- Si può ricodificare l'output y con un vettore $\mathbf{y}$ K-dimensionale, dove tutte le entrate sono uguali a $-\frac{1}{K-1}$ eccetto a 1 in posizione k se $y{=}k$, cioè $\mathbf{y} = (y_1, ..., y_K)^t$ e:

$$y_k = \begin{cases} 1 & y = k \\ -\frac{1}{K-1} & y \neq k \end{cases} \tag{2}$$

- La generalizzazione della funzione loss esponenziale al caso multivariato segue naturalmente:

$$L(\mathbf{y},\mathbf{f}) = exp(-\frac{1}{K}(y_1 f_1 + ... + y_K f_K)) = (-\frac{1}{K}\mathbf{y}^{\mathbf{t}}\mathbf{f}) \tag{3}$$

Giustificazione teorica

- Si è interessati a:

$$\underset{\mathbf{f}(\mathbf{x})}{\text{argmin}}\ \mathrm{E}_{\mathbf{Y}|\mathbf{x}} \exp\left(-\frac{1}{K}(Y_1 f_1 + ... + Y_K f_K)\right)$$

$$\text{soggetto al vincolo } f_1 + ... + f_K = 0$$

- Attraverso il metodo dei moltiplicatori di Lagrange questo problema di ottimizzazione vincolata può esser riscritto come:

$$exp(-\frac{f_1(\mathbf{x})}{K-1} Prob((y) = 1|\mathbf{x})) + \cdots + exp(-\frac{f_K(\mathbf{x})}{K-1} Prob((y) = K|\mathbf{x})) +$$
$$-\lambda(f_1(\mathbf{x}) + \cdots + f_K(\mathbf{x})$$

dove λ è il moltiplicatore di Lagrange.

Giustificazione teorica

- Si calcolano le derivate rispetto a f_k e λ e le si pone uguali a 0.
- Risolvendo questo sistema di equazioni, si ottiene:

$$f_k^*(\mathbf{x}) = (K-1)(logProb(y=k|\mathbf{x}) - \frac{1}{K}\sum_{k'=1}^{K} logProb(y=k'|\mathbf{x})) \quad (4)$$

  con k=1,...,K
- Perciò:

$$\underset{k}{\text{argmax}} f_k^*(\mathbf{x}) = \underset{k}{\text{argmax}} Prob(y=k|\mathbf{x})$$

  che è la regola di classificazione ottimale di Bayes per l'errore.
- Questo giustifica l'uso della funzione loss esponenziale multiclasse. Inoltre, l'algoritmo SAMME è equivalente al modello additivo per forward stagewise utilizzando la loss function esponenziale multiclasse trovata.

Adaboost.MH

- Sia Y un set finito di etichette, e sia $k = |Y|$.
- Nel caso multi-label, ogni istanza $x \in X$ può appartenere a diverse classi in Y. Quindi, un esempio etichettato è una coppia $(x, \mathcal{Y})$ dove $\mathcal{Y} \subseteq Y$ nel set di etichette assegnate a x.
- Lo scopo è di predire tutte e solo le etichette corrette. Ovvero, l'algoritmo di apprendimento genera un'ipotesi che predice set di etichette, e la loss dipende su come queste predizioni differiscono da una che è stata osservata.

Adaboost.MH

- Perciò, $H : X \rightarrow 2^Y$ e, rispetto alla distribuzione D la loss è:
  $$\frac{1}{k} E_{(x,Y)\sim D}\left[|h(x) \triangle \mathcal{Y}|\right]$$
  dove $\triangle$ denota la differenza simmetrica (il valore 1/k è utilizzato solo per assicurarsi che il valore stia in $[0, 1]$).
- Chiamiamo questa misura *Hamming loss* di H, e la denotiamo come $hloss_D(H)$.
- Per minimizzare la Hamming loss, si può, in maniera naturale, decomporre il problema in k problemi di classificazione binaria ortogonali. Si pensa a $\mathcal{Y}$ come k etichette binarie (dipendendo da che un'etichetta y sia o non sia inclusa in $\mathcal{Y}$). Similmente, *h(x)* può essere vista come k predizioni binarie.

Adaboost.MH

- La Hamming loss poi può essere considerata come una media dell'errore di h su questi k problemi binari.
- Per $\mathcal{Y} \subseteq Y$, si definisce $\mathcal{Y}[l]$ per $l \in Y$ essere:

$$\mathcal{Y}[l] = \begin{cases} 1 & l \in \mathcal{Y} \\ -1 & l \notin \mathcal{Y} \end{cases}$$

- L'idea principale della riduzione è semplicemente ripetere ogni esempio di training (x_i, Y_i) per k esempi $((x_i, l), Y_i[l])$ per $l \in \mathcal{Y}$.

Adaboost.MH

- Dati: $(x_1, \mathcal{Y}_1), ..., (x_m, \mathcal{Y}_m)$ dove $x_i \in X, \mathcal{Y}_i \subseteq Y$
- Si inizializza $D_1(i, l) = 1/(mk)$
- Per t = 1, ..., T

1. Viene chiamato il weak learner utilizzando la distribuzione D_t
2. Il weak learner, in risposta, genera un'ipotesi $h_t : X \times Y \to \mathbb{R}$
3. Viene calcolato $\alpha_t \in \mathbb{R}$
4. Viene aggiornata:

$$D_{t+1}(i, l) = \frac{D_t(i,l) exp(-\alpha_t Y_i[l] h_t(x_i,l))}{Z_t}$$

   dove Z_t è un fattore di normalizzazione

L'output sarà:

$$h_{fin}(x, l) = sign(\sum_{t=1}^{T} \alpha_t h_t(x, l))$$

Conclusioni

- Sono state analizzate alcune varianti del filtro Adaboost nel caso multiclasse.
- Per quanto riguarda lo studio effettuato durante il tirocinio, sebbene il calcio abbia il concetto multi-label, è stato scelto di utilizzare l'Adaboost.M2.
- Questo perchè ho dovuto evitare la fase di addestramento, avendo già i bookmakers che sono stati considerato come i classificatori weak istruiti.
- È stato mio compito quello di modificare il programma che implementa l'algoritmo per riconoscere i bookmakers come suoi classificatori deboli e non crearne di nuovi.