UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Margini ale erorilor la generalizare pentru diferiți algoritmi de învățare automată

propusă de

Cristian Vîntur

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Liviu Ciortuz

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Margini ale erorilor la generalizare pentru diferiți algoritmi de învățare automată

Cristian Vîntur

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Liviu Ciortuz

Avizat,

Îndrumător Lucrare

Titlul, Numele și prenumele Conf. Dr. Liviu Ciortuz

Data: ..... Semnătura: .....

DECLARAȚIE

privind originalitatea conținutului lucrării de licență

Subsemnatul **Vîntur Cristian** domiciliul în **România, jud. Iași, mun. Pașcani, strada Cuza Vodă, nr. 12, bl. D10, et. 2, ap. 8**, născut la data de **09 februarie 1997**, identificat prin CNP **1970209271545**, absolvent al Universității „Alexandru Ioan Cuza" din Iași, Facultatea de informatică, specializarea **informatică**, promoția 2019, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Margini ale erorilor la generalizare pentru diferiți algoritmi de învățare automată** elaborată sub îndrumarea domnului **Conf. Dr. Liviu Ciortuz**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi: ..... Semnătură student: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Margini ale erorilor la generalizare pentru diferiți algoritmi de învățare automată**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Cristian Vîntur**

Data: ..... Semnătura: .....

Cuprins

Motivație

Domeniul învățării automate s-a dezvoltat foarte mult în ultimii ani și și-a găsit aplicații în afara informaticii, precum în transport, finanțe sau medicină. De exemplu, problemele identificării tipului de sol dintr-o fotografie făcută din satelit sau diferențierea între bancnote reale și falsificate pot fi rezolvate cu algoritmi de clasificare precum cei prezentați în primele două capitole.

Deși cel mai important aspect al unui algoritm de învățare automată este acuratețea în practică, obținerea de margini superioare pentru erorile la generalizare pot fi de folos în explicarea comportamentului algoritmului sau alegerea hiper-parametrilor. În această lucrare vom oferi câteva rezultate teoretice care mărginesc această eroare nu în funcție de numărul de instanțe de antrenament clasificate corect, ci în funcție de confidența cu care acestea au fost clasificate.

Contribuții

Pentru a putea prezenta rezultatele teoretice din această lucrare a fost necesară o cercetare pe termen lung, timp în care m-am familiarizat cu multe concepte noi și care au o aplicabilitate practică pe scară largă. Faptul că am prezentat unele noțiuni pe care articolele din care este inspirată această lucrare doar le enunță, ajută cititorul să înțeleagă mai bine modul în care aceste teoreme pot fi folosite în practică. De asemenea, acolo unde a fost posibil, am încercat să detaliez anumiți pași din demonstrații care au o trecere bruscă de la o relație la alta.

Pe partea de experimente, am implementat algoritmul AdaBoost în 3 variante, precum și algoritmul propus în [5] fără regularizare. Acești algoritmi au fost rulați pe diferite seturi de date pentru a obține o imagine generală a comportamentului lor.

Introducere

Capitolele vor fi structurate în felul următor: voi începe printr-o scurtă prezentare a algoritmilor, urmată de prezentarea unor rezultate teoretice, iar la final o discuție pe baza acestor rezultate.

Primul capitol începe printr-o scurtă introducere în conceptul de *boosting*, precum și într-un algoritm care folosește această idee, propus de Yoav Freund și Robert Schapire în 1997, care se numește AdaBoost. Voi expune principalele idei de implementare și voi prezenta două teoreme care să explice eficiența acestui algoritm în practică.

În capitolul 2 voi prezenta un alt algoritm de clasificare bazat pe regresie logistică care, spre deosebire de AdaBoost, funcționează pe cazul *multiclass*. Voi enunța diferite tipuri de clasificatori care pot fi utilizați de acest algoritm și voi termina cu o teoremă care să ofere o margine superioară pentru eroarea la generalizare.

Capitolul 3 introduce un nou tip de problemă, asemănătoare cu ceea ce este cunoscut în literatură ca *learning from logged bandit feedback* [10]. Voi descrie pe scurt o abordare propusă de *Shalit* și *Johansson* și voi prezenta o teoremă care mărginește superior un tip diferit de eroare la generalizare definit ulterior.

În capitolul 4 voi rula algoritmii prezentați pe câteva seturi de date reale pentru a observa și eficiența lor în practică. Seturile de date au fost alese cât mai variate, atât după mărime, cât și după distribuția instanțelor.

Capitolul 1

Boosting

1.1 Introducere

În acest capitol ne vom ocupa de algoritmi de învățare automată supervizată care etichetează o anumită instanță combinând predicțiile altor clasificatori (ipoteze). Această metodă poartă numele de *Boosting*. Scopul algoritmului este acela de a crea un clasificator care are eroare la generalizare mică, adică eroarea pe o mulțime de instanțe necunoscute în timpul antrenamentului.

Considerăm că fiecare instanță este reprezentată de d atribute - numere reale. Notăm cu $X = \mathbb{R}^d$ spațiul instanțelor și cu Y mulțimea tuturor etichetelor posibile. Ne vom concentra pe algoritmi de clasificare binară, adică pe cazurile în care $|Y| = 2$. Pentru simplitatea notațiilor și a calculelor, vom considera că $Y = \{-1, 1\}$.

În continuare, dacă nu este precizat altfel, prin x vom înțelege o instanță oarecare din X, iar prin (x, y) vom înțelege o instanță oarecare din X a cărei etichetă corectă este $y \in Y$. Dat fiind un clasificator f, eticheta prezisă pentru o instanță $x \in X$ este $sign\,(f(x))$. Se observă că f etichetează corect instanța (x, y) dacă și numai dacă $yf(x) > 0$.

Definim marginea unei instanțe (x, y), pe cazul general, ca fiind diferența dintre suma ponderilor tuturor clasificatorilor care etichetează instanța x cu y și suma maximă a ponderilor clasificatorilor care o etichetează cu $\widetilde{y}$, unde maximul se face după orice $\widetilde{y} \neq y$ [1]. Formal, dacă ipotezele combinate de algoritmul de *boosting* sunt $h_1, h_2, \ldots, h_t$ având ponderile respectiv $w_1, w_2, \ldots, w_t$, atunci

$$margin(x) = \sum_{h_i(x)=y} w_i - \max_{\substack{\widetilde{y} \in Y \\ \widetilde{y} \neq y}} \left(\sum_{h_i(x)=\widetilde{y}} w_i \right) \in [-1, 1]$$

În cazul în care $Y = \{-1, 1\}$, se arată prin calcul direct că marginea instanței (x, y) este $yf(x)$. Așa cum era de așteptat, am ajuns la concluzia că o instanță oarecare este clasificată corect dacă și numai dacă marginea ei este pozitivă. Mai mult, cu cât marginea este mai mare, cu atât algoritmul este mai încrezător că instanța este clasificată corect.

1.2 Clasificatori slabi

În contextul algoritmului AdaBoost ne vom ocupa de clasificarea binară. Astfel, un clasificator care etichetează fiecare instanță în mod aleator și uniform are eroarea la antrenare de $1/2$.

Un clasificator slab este, prin definiție, orice algoritm de clasificare care are eroarea la antrenare strict mai mică decât cea a algoritmului de mai sus: $err_D(A) < 1/2$. Eroarea este una ponderată, ea depinzând atât de mulțimea de instanțe, cât și de o distribuție (pondere) peste această mulțime. În experimente, clasificatorii slabi considerați sunt arbori de decizie de înălțime 1.

1.3 Dimensiunea Vapnik-Chervonenkis

Dimensiunea Vapnik-Chervonenkis este o măsură a complexității a spațiului de funcții care pot fi învățate de un algoritm de clasificare. Este definit ca numărul maxim de puncte care pot fi separate de algoritm.

Pentru un set de puncte dat, spunem că algoritmul le poate separa dacă, pentru orice atribuire de etichete posibilă, există un clasificator care poate fi învățat și care să clasifice corect toate aceste puncte.

Formal, fie punctele $x_1, x_2, \cdots, x_d$ și C mulțimea tuturor clasificatorilor care pot fi învățați de algoritm. Punctele pot fi separate dacă și numai dacă:

$$|\{(f(x_1), f(x_2), \cdots, f(x_d)) \mid f \in C\}| = 2^d$$

1.4 Scurtă introducere în algoritmul AdaBoost

AdaBoost construiește un clasificator combinând mai mulți clasificatori slabi, asociind fiecăruia câte o pondere. Algoritmul este adaptiv în sensul că, la fiecare iterație,

distribuția asupra instanțelor se modifică, de unde provine și numele: *Adaptive Boosting*. Algoritmul mărește ponderea asociată instanțelor clasificate greșit, în același timp scăzând ponderea asociată instanțelor clasificate corect. Efectul este acela că următorii clasificatori se vor concentra pe acele instanțe mai greu de clasificat.

În continuare prezentăm pseudocodul algoritmului:

intrare: S - mulțimea instanțelor de antrenament, $|S| = m$

A - un algoritm care primește la intrare o mulțime de instanțe și o distribuție peste această mulțime și produce un clasificator slab

T - numărul de iterații

ieșire : F - o funcție definită pe X cu valori în {-1, 1}

inițializare: $D_i^1 = \frac{1}{m}$

for $t \leftarrow 1$ **to** T **do**
- Rulează algoritmul A folosind distribuția D_t, obținând clasificatorul slab h_t
- Calculează eroarea la antrenare corespunzătoare clasificatorului h_t și distribuției D_t : $\epsilon \leftarrow err_{D_t}(h_t)$
- Calculează ponderea asociată ipotezei h_t : $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon}$
- Calculează noua distribuție D_{t+1}
- **for** $i \leftarrow 1$ **to** m **do**
  - $D_i^{t+1} = \frac{1}{Z_t} \cdot D_i^t \cdot \exp\left(\alpha_t h_t\left(x_i\right) y_i\right)$
- **end**
- unde Z_t este factor de normalizare astfel încât $\sum_{i=1}^{m} D_i^{t+1} = 1$
- $F = F + h_t$

end

return F

Algorithm 1: AdaBoost

Deși nu acesta era scopul inițial, s-a descoperit că AdaBoost minimizează la fiecare iterație funcția de cost exponențială

$$L(h(x), y) = \exp\left(-h(x)y\right)$$

Algoritmul se poate generaliza pentru a minimiza, în mod *greedy*, și alte funcții de cost mai generale. De obicei, pentru a putea găsi eficient clasificatorul slab cu eroarea ponderată minimă și pentru a evita punctele de minim local, se lucrează cu funcții derivabile și convexe. Pentru experimente, am implementat atât varianta clasică prezentată

în pseudocod, cât și alte două variante care minimizează funcțiile de cost logistică

$$L(h(x), y) = \ln\left(1 + \exp\left(-h(x)y\right)\right)$$

și pătratică

$$L(h(x), y) = \left(h(x) - y\right)^2$$

1.5 Detalii de implementare

Atât algoritmul AdaBoost, cât și clasificatorii slabi folosiți, au fost implementați de mine în C++, folosind biblioteci STL. Cu excepția faptului că versiunea care trebuie instalată este C++17, nu sunt necesare alte biblioteci / programe.

Datele de antrenament sunt stocate ca o matrice, fiecărei instanțe corespunzându-i un rând, iar etichetele instanțelor (± 1) fiind pe ultima coloană.

Clasa pentru clasificatorul slab conține următoarele variabile: *dim*, *threshold*, *label*, *alpha*. Semnificația este următoarea: fiind dată o instanță $x \in \mathbb{R}$, clasificatorul va returna *label* dacă $x_{dim} \leq threshold$ și $-$*label* în caz contrar. Parametrul *alpha* reprezintă ponderea asociată de algoritmul AdaBoost.

Precizăm acum mulțimea clasificatorilor slabi din care va alege algoritmul. Pentru fiecare dimensiune i din cele d ale instanțelor, considerăm $x_{1i} \leq x_{2i} \leq \cdots \leq x_{mi}$ valorile de pe dimensiunea i. Pentru fiecare $x_{ji} \neq x_{j+1,i}$ considerăm doi clasificatori slabi, având $threshold = \left(x_{ji} + x_{j+1,i}\right)/2$, iar $label = \pm 1$. De asemenea, mai considerăm doi clasificatori *extremi*, care clasifică toate instanțele la fel.

La fiecare iterație, algoritmul calculează eroarea ponderată a fiecărui clasificator considerat mai sus și îl reține pe acela cu eroarea minimă. Complexitatea acestui algoritm este $O\left(dm^2\right)$ pe iterație deoarece sunt $O\left(dm\right)$ clasificatori în total, iar calculul erorii se face în $O\left(m\right)$. Pentru a reduce complexitatea, am folosit următoarea idee: am sortat, la început, fiecare coloană din matricea de valori (adică, pentru fiecare dimensiune, sortăm valorile corespunzătoare). În acest moment, calculul erorilor tuturor clasificatorilor de pe o anumită dimensiune se face în timp liniar folosind o simplă parcurgere, ținând cont că, la fiecare pas se modifică eticheta unei singure instanțe. Complexitatea finală este $O\left(tdm\right)$, cu $O\left(dm\ log(m)\right)$ preprocesare, unde t reprezintă numărul de iterații.

Deși calculele necesare pentru a găsi clasificatorul slab optim și ponderea acestuia care minimizează funcția de cost logistică sunt mai complicate (detalii în [2]), singura

diferență față de varianta clasică este relația pentru actualizarea distribuției:

$$D_i^{t+1} = \frac{1}{Z_t} \cdot \frac{1}{1 + \exp\left(y_i F_{t-1}\left(x_i\right)\right)}$$

Pentru funcția de cost pătratică, expresia de minimizat este:

$$\sum_{i=1}^{m} \left(f_t\left(x_i\right) - y_i\right)^2 = \sum_{i=1}^{m} \left(\alpha_t h_t\left(x_i\right) - r_i\right)^2$$

unde $r_i = y_i - f_{t-1}\left(x_i\right)$. Aceasta este o ecuație de gradul 2 în α_t având minimul în

$$\widehat{\alpha}_t = \frac{\sum_{i=1}^{m} r_i h_t\left(x_i\right)}{\left\|h_t\right\|^2}$$

Deoarece $h_t\left(x_i\right) \in \{-1, 1\}$, rezultă $\left\|h_t\right\|^2 = m = const$, deci e suficient să minimizăm expresia de la numărător. Acest lucru se poate face în mod eficient precalculând reziduurile r_i și folosind tehnica sortării datelor prezentată la varianta clasică.

În continuare prezentăm două teoreme, preluate din [1], care mărginesc eroarea la generalizare a clasificatorilor obținuți prin *boosting* în funcție de distribuția marginilor și a unor parametri legați de mulțimea de antrenament și mulțimea H a clasificatorilor de bază. Prima din oferă o margine superioară în cazul în care H este finită, iar a doua folosește dimensiunea Vapnik-Chervonenkis atunci când H este infinită.

1.6 O margine superioară în cazul unui număr finit de clasificatori de bază

Notăm cu H mulțimea tuturor clasificatorilor de bază. Considerăm înfășurătoarea convexă a lui H definită ca mulțimea tuturor mediilor ponderate a elementelor din H:

$$C = \left\{ f : x \to \sum_{h \in H} a_h h(x) \middle| a_h \geq 0; \sum_{h \in H} a_h = 1 \right\}$$

Teorema 1. Fie D o distribuție peste $X \times \{-1, 1\}$ și S o mulțime de m elemente alese independent conform distribuției D. Presupunem că mulțimea H a clasificatorilor de bază este finită și fie $\delta > 0$. Atunci, cu o probabilitate de cel puțin $1 - \delta$ peste alegerea lui S, fiecare clasificator $f \in F$ satisface următoarea inegalitate, pentru orice $\theta > 0$:

$$P_D\left(yf(x) \leq 0\right) \leq P_S\left(yf(x) \leq \theta\right) + O\left(\frac{1}{\sqrt{m}}\left(\frac{\log m \cdot \log |H|}{\theta^2} + \log\frac{1}{\delta}\right)^{\frac{1}{2}}\right)$$

Demonstrație

Pentru orice $N \geq 1$ natural, notăm cu

$$C_N = \left\{ f : x \to \frac{1}{N} \sum_{i=1}^{N} h_i(x) \,\middle|\, h_i \in H \right\}$$

mulțimea tuturor mediilor a N funcții din H (o funcție poate apărea de mai multe ori). Pe măsură ce N crește, această mulțime aproximează din ce în ce mai bine mulțimea C.

Fie $f = \sum_{h \in H} a_h h, \quad \sum_h a_h = 1$ un clasificator oarecare. Considerăm distribuția A peste H corespunzătoare coeficienților a_h. Asociem lui f o distribuție Q peste C_N astfel: un element $g \in C_N$ este ales aleator conform lui Q alegând N clasificatori de bază $g_i \in H$ conform lui A și făcând media lor: $g = \frac{1}{N} \sum_{i=1}^{N} g_i$.

Fie G evenimentul aleator corespunzător alegerii lui g și G_i evenimente aleatoare și independente corespunzătoare alegerii lui g_i.

$$E[G_i] = \sum_{h \in H} a_h h = f, \forall 1 \leq i \leq N$$

$$E[G] = E\left[\frac{1}{N} \sum_{i=1}^{N} G_i\right] = \frac{1}{N} \sum_{i=1}^{N} E[G_i] = \frac{1}{N} \sum_{i=1}^{N} f = f \tag{1}$$

Folosind faptul că, în general, $P(X \wedge Y) \leq P(Y)$ unde X și Y sunt două evenimente aleatoare următoarea inegalitate este adevărată:

$$\begin{aligned}
P_D[yf(x) \leq 0] &= 1 \cdot P_D[yf(x) \leq 0] = \left(\sum_{g \in C_N} P_Q(g) \right) P_D[yf(x) \leq 0] \\
&= \sum_{g \in C_N} P_Q(g) \cdot (P_D[yf(x) \leq 0 \wedge yg(x) \leq \theta/2] + P_D[yf(x) \leq 0 \wedge yg(x) > \theta/2]) \\
&\leq \sum_{g \in C_N} P_Q(g) \cdot (P_D[yg(x) \leq \theta/2] + P_D[yf(x) \leq 0 \wedge yg(x) > \theta/2]) \\
&= \sum_{g \in C_N} P_Q(g) \cdot P_D[yg(x) \leq \theta/2] + \sum_{g \in C_N} P_Q(g) \cdot P_D[yf(x) \leq 0 \wedge yg(x) > \theta/2]
\end{aligned}$$

Rezultă:

$$P_D[yf(x) \leq 0] \leq E_{g \sim Q}[P_D[yg(x) \leq \theta/2]] + \sum_{g \in C_N} P_Q(g) \cdot P_D[yf(x) \leq 0 \wedge yg(x) > \theta/2] \tag{2}$$

Majorăm al doilea termen al inegalității (2). Folosind definiția funcției de proba-

bilitate, avem:

$$\begin{aligned}
&\sum_{g\in C_N} P_Q(g)\cdot P_D[yf(x)\le 0 \wedge yg(x)>\theta/2] \\
&\le \sum_{g\in C_N} P_Q(g)\cdot P_D[yg(x)>\theta/2|yf(x)\le 0] \\
&= \sum_{g\in C_N} P_Q(g)\sum_{(x,y)} P_D[x,y]\cdot[[yg(x)>\theta/2|yf(x)\le 0]] \\
&= \sum_{(x,y)} P_D[x,y]\sum_{g\in C_N} P_Q(g)\cdot[[yg(x)>\theta/2|yf(x)\le 0]] \\
&= \sum_{(x,y)} P_D[x,y]P_Q[yg(x)>\theta/2|yf(x)\le 0] \\
&= E_{(x,y)\sim D}\left[P_Q[yg(x)>\theta/2|yf(x)\le 0]\right]
\end{aligned}$$

Din relația (1) avem $E[G]=f \Leftrightarrow E[yG]=yf$, deci

$$E_Q[yg(x)|yf(x)\le 0]\le 0$$

Aplicând teorema lui Chernoff [3], obținem următoarea margine superioară:

$$P_Q[yg(x)>\theta/2|yf(x)\le 0]\le \exp\left(-\frac{N\theta^2}{8}\right)$$

Deci:

$$E_{(x,y)\sim D}\left[P_Q[yg(x)>\theta/2|yf(x)\le 0]\right]\le \exp\left(-\frac{N\theta^2}{8}\right) \tag{3}$$

Majorăm acum primul termen al inegalității (2). Probabilitatea ca, alegând aleator mulțimea S,

$$P_D[yg(x)\le\theta/2] > P_S[yg(x)\le\theta/2]+\epsilon_N$$

pentru $g\in C_N, \theta>0, \epsilon_N>0$ fixate este, conform teoremei lui Chernoff, cel mult $\exp(-2m\epsilon_N^2)$. Probabilitatea ca pentru $\epsilon_N>0$ fixat să existe măcar un $g\in C_N$ și un $\theta>0$ pentru care inegalitatea de mai sus să fie adevărată este cel mult

$$(N+1)\,|C_N|\exp\left(-2m\epsilon_N^2\right).$$

$|C_N|$ reprezintă numărul de posibilități de a alege funcția g, iar $N+1$ reprezintă numărul de posibilități de a alege θ: deoarece $yg(x)\in\left\{-\frac{N}{N},\ldots,\frac{0}{N},\ldots,\frac{N}{N}\right\}$, rezultă că este suficient să alegem θ de forma $\frac{2i}{N}, 0\le i\le N$. Se observă ușor că $|C_N|\le|H|^N$.

Pentru

$$\epsilon_N=\sqrt{\frac{1}{2m}\ln\frac{(N+1)\,|H|^N}{\delta_N}}$$

obținem că, cu probabilitate de cel puțin $1 - \delta_N$, avem:

$$P_D[yg(x) \leq \theta/2] \leq P_S[yg(x) \leq \theta/2] + \epsilon_N$$
$$P_{D,g\sim Q}[yg(x) \leq \theta/2] \leq P_{S,g\sim Q}[yg(x) \leq \theta/2] + \epsilon_N \qquad (4)$$

deoarece prima inegalitate are loc pentru orice $g \in C_N$ și $\theta > 0$.

$$\begin{aligned} P_{S,g\sim Q}[yg(x) \leq \theta/2] &\leq P_{S,g\sim Q}[yf(x) \leq \theta] + P_{S,g\sim Q}[yg(x) \leq \theta/2, yf(x) > \theta] \\ &= P_S[yf(x) \leq \theta] + E_S\left[P_{g\sim Q}[yg(x) \leq \theta/2, yf(x) > \theta]\right] \\ &\leq P_S[yf(x) \leq \theta] + E_S\left[P_{g\sim Q}[yg(x) \leq \theta/2 \big| yf(x) > \theta]\right] \qquad (5) \end{aligned}$$

Majorăm probabilitatea din interiorul mediei folosind, din nou, teorema lui Chernoff:

$$E_S\left[P_{g\sim Q}[yg(x) \leq \theta/2 \big| yf(x) > \theta]\right] \leq e^{-\frac{N\theta^2}{8}} \qquad (6)$$

Fie $\delta_N = \frac{\delta}{N(N+1)}$. Probabilitatea ca inegalitatea (4) să fie falsă pentru un N fixat este cel mult δ_N. Probabilitatea ca inegalitatea (4) să fie falsă pentru cel puțin un N este cel mult $\sum_{N\geq 1} \delta_N = \delta$.

Combinând relațiile (2), (3), (4), (5) și (6), rezultă că, cu probabilitate mai mare sau egală decât $1-\delta$, următoarea inegalitate este adevărată pentru orice $\theta > 0$ și $N \geq 1$:

$$P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + 2\exp\left(-\frac{N\theta^2}{8}\right) + \sqrt{\frac{1}{2m}\ln\frac{N\left(N+1\right)^2|H|^N}{\delta}}$$

Teorema rezultă pentru

$$N = \left\lceil \frac{4}{\theta^2}\ln\frac{m}{\ln|H|} \right\rceil$$

1.7 O margine superioară în cazul unui număr infinit de clasificatori de bază

Teorema 2. Fie D o distribuție peste $X \times \{-1, 1\}$ și S o mulțime de m elemente alese independent conform distribuției D. Presupunem că mulțimea H a clasificatorilor de bază are dimensiune Vapnik Chervonenkis d și fie $\delta > 0$. Presupunem că $m \geq d \geq 1$. Atunci, cu o probabilitate de cel puțin $1 - \delta$ peste alegerea lui S, fiecare clasificator $f \in F$ satisface următoarea inegalitate, pentru orice $\theta > 0$:

$$P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \delta] + O\left(\frac{1}{\sqrt{(}m)}\left(\frac{\log m \cdot \log H}{\theta^2} + \log\frac{1}{\delta}\right)^{\frac{1}{2}}\right)$$

Demonstrație

Demonstrația utilizează următorul rezultat de convergență uniformă datorat lui Devroye. Pentru A o mulțime de submulțimi ale unui spațiu Z, notăm:

$$s\left(A,m\right)=max\left\{\left|\left\{A\cap S:A\in A\right\}\right|:S\subseteq Z,\left|S\right|=m\right\}$$

Lemă (Devroye). Pentru orice mulțime A de submulțimi ale lui Z și pentru o mulțime S de m elemente alese aleator din Z conform unei distribuții D, avem:

$$P_{S\sim D^m}\left[\sup_{A\in A}\left|P_{z\sim S}\left[z\in A\right]-P_{z\sim D}\left[z\in A\right]\right|>\epsilon\right]\leq 4e^8 s\left(A,m^2\right)e^{-2m\epsilon^2}$$

Cu alte cuvinte, lema majorează probabilitatea unei diferențe semnificative între probabilitățile empirice și reale ale oricărui eveniment din A.

Revenind la demonstrația teoremei 2, procedăm analog până la majorarea primului termen din inegalitatea (2). În loc să folosim inegalitatea lui Boole, folosim lema lui Devroye.

Fie

$$A=\left\{\left\{(x,y)\in X\times\{-1,1\}:yg(x)>\theta/2\right\}:g\in C_N,\theta>0\right\}$$

Fie $x_1,x_2,\ldots,x_m\in X$ și $y_1,y_2,\ldots,y_m\in\{-1,1\}$. Deoarece dimensiunea VC a lui H este d, lema lui Sauer [3] spune că:

$$\left|\left\{\langle h(x_1),h(x_2),\ldots,h(x_m)\rangle:h\in H\right\}\right|\leq\sum_{i=0}^{d}\binom{m}{i}\leq\left(\frac{em}{d}\right)^d$$

pentru $m\geq d\geq 1$. Această relație implică:

$$\left|\left\{\langle y_1g(x_1),y_2g(x_2),\ldots,y_mg(x_m)\rangle:g\in C_N\right\}\right|\leq\left(\frac{em}{d}\right)^{dN}$$

deoarece fiecare $g\in C_N$ este compus din N funcții din H. Folosind același argument ca la teorema anterioară, este suficient să considerăm doar $N+1$ valori distincte pentru θ, deci $s\left(A,m\right)\leq\left(N+1\right)\left(em/d\right)^{dN}$. Acum aplicăm lema lui Devroye pentru a majora termenul din interiorul mediei al inegalității (2). Fixând

$$\epsilon_N=\sqrt{\frac{1}{2m}\left(dN\ln\left(\frac{em^2}{d}\right)+\ln\left(\frac{4e^8\left(N+1\right))}{\delta_N}\right)\right)}$$

și luând media conform distribuției Q, obținem că, cu probabilitate de cel puțin $1-\delta_N$, (4) este adevărată pentru orice $\theta>0$. Continuând la fel ca la teorema precedentă, obținem că, cu probabilitate de cel puțin $1-\delta$, pentru orice $\theta>0$ și $N\geq 1$:

$$\begin{aligned}P_D[yf(x)\leq 0]\leq & P_S[yf(x)\leq\theta]+2\exp\left(-\frac{N\theta^2}{8}\right)\\&+\sqrt{\frac{1}{2m}\left(dN\ln\left(\frac{em^2}{d}\right)+\ln\left(\frac{4e^8N\left(N+1\right))^2}{\delta}\right)\right)}\end{aligned}$$

Teorema rezultă pentru

$$N = \left\lceil \frac{4}{\theta^2} \ln \frac{m}{d} \right\rceil$$

1.8 Discuții

Prima dată voi prezenta o problemă practică care poate fi rezolvată folosind aceste teoreme, iar la final voi arăta modul în care aceste teoreme explică faptul binecunoscut că, în general, algoritmul AdaBoost nu produce *overfitting*.

Considerăm următoarea problemă: avem o mulțime de date pe care trebuie să antrenăm un clasificator, dar nu avem o mulțime date pe care să testăm clasificatorul rezultat. Cum ne putem asigura că modelul va prezice corect etichetele unor instanțe noi? Nu putem testa modelul tot pe datele de antrenament, pentru că nu am putea ști că nu s-a produs fenomenul de *overfitting*.

O metodă uzuală care rezolvă această problemă se numește *k-fold cross-validation*: se partiționează datele de antrenament în k mulțimi de dimensiuni aproximativ egale, iar pentru fiecare din aceste mulțimi M_i:

- consideră M_i ca fiind mulțime de testare
- antrenează un clasificator folosind celelalte $k-1$ mulțimi
- calculează eroarea clasificatorului pe mulțimea M_i.

Pentru $k = n$, numărul tuturor instanțelor, obținem *leave-one-out cross-validation*.

O altă metodă se bazează pe rezultatele din secțiunile 1.6 și 1.7. Antrenăm un clasificator pe mulțimea de antrenament pe care o avem și calculăm marginile tuturor instanțelor. Dacă există foarte puține instanțe pentru care aceste margini sunt *mici*, atunci putem argumenta că, foarte probabil, clasificatorul se va comporta bine și pe instanțe noi. Avantajul acestei metode este acela că antrenarea poate fi efectuată folosind tot setul inițial de date, efectul fiind obținerea unui clasificator mai bun.

Arătăm acum modul în care teoremele caracterizează comportamentul algoritmului AdaBoost. Observația principală este aceea că printre parametrii care apar în expresiile pentru marginea superioară nu se regăsește numărul de iterații (care coincide cu numărul de clasificatori slabi utilizați de schema de votare). Cu alte cuvinte, eroarea la generalizare nu are tendința să crească pe măsură ce modelul devine din ce în ce mai complex, producând efectul de *overfitting*. Din păcate, pentru a oferi o

margine rezonabilă, numărul de instanțe de antrenament trebuie să fie foarte mare, de ordinul zecilor de mii.

Capitolul 2

Sparse multinomial logistic regression

2.1 Introducere

În acest capitol vom prezenta un algoritm de clasificare $multiclass$, adică în care mulțimea tuturor etichetelor posibile poate avea cardinal mai mare decât 2 (dar trebuie să fie finită).

Ca și în capitolul precedent, vom reprezenta o instanță print-un vector $x \in \mathbb{R}^d$ de dimensiune d. Pentru reprezentarea etichetelor vom adopta o altă notație. Fie n numărul de etichete posibile. Eticheta unei instanțe x va fi reprezentată de un vector $y \in \{0,1\}^n$ unde $y_i = 1$ dacă și numai dacă x aparține celei de-a i-a clase.

Scopul algoritmului este de a învăța un vector de vectori w (câte unul pentru fiecare din cele n etichete posibile), astfel încât eroarea la generalizare să fie cât mai mică. Folosind notațiile din paragraful precedent, probabilitatea ca instanța (x, y) să aparțină clasei i este [5]:

$$p\left(y_i = 1 \mid x, w\right) = \frac{\exp\left(w_i^T x\right)}{\sum_{j=1}^{n} \exp\left(w_j^T x\right)}$$

unde w_i reprezintă vectorul corespunzător clasei i. Deoarece suma probabilităților, făcută peste toate clasele, este 1, unul din vectori nu este necesar să fie estimat. Fixăm $w_n = 0$ și înțelegem prin w un vector $d(n-1)$-dimensional, reprezentând concatenarea celor $n-1$ vectori corespunzători primelor $n-1$ clase.

2.2 Divergența Kullback-Leibler

Divergența Kullback-Leibler este o măsură a distanței între două probabilități. Divergența a două probabilități p și q se notează prin $D_{KL}\left(p \mid\mid q\right)$ și, în cazul discret,

este definită prin:

$$D_{KL}\left(p \mid\mid q\right) = -\sum_{x} p(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Pentru variabile aleatoare continue, definiția ei este următoarea:

$$D_{KL}\left(p \mid\mid q\right) = -\int_{-\infty}^{\infty} p(x) \log\left(\frac{q(x)}{p(x)}\right) dx$$

Divergența KL este definită doar în cazul în care $q(x) = 0$ implică $p(x) = 0$, termenul corespunzător fiind interpretat prin 0.

Deoarece divergența KL reprezintă o măsură a distanței, o valoare mai mică a acesteia semnifică o apropiere mai mare între cele două distribuții.

2.3 Tipuri de clasificatori

Teorema din secțiunea 2.5 funcționează în cazul clasificării binare. Ca și la AdaBoost, considerăm că $Y = \{-1, 1\}$.

Algoritmul prezentat returnează un vector $\widehat{w}$. Considerăm de asemenea și o distribuție oarecare q peste mulțimea tuturor vectorilor w. Prezentăm 3 tipuri de clasificatori [5]:

- *Point Classifier (PC)*. Acest clasificator returnează o etichetă după regula $f_{PC}(x) = sign\left(\widehat{w}^T x\right)$.
- *Bayes Voting Classifier (BVC)*. Acest clasificator folosește o schemă de votare bazată pe distribuția q: $f_{BVC}(x, q) = sign\left(E_q\left[sign\left(w^T x\right)\right]\right)$.
- *Gibbs Classifier (GC)*. Acest clasificator alege un vector w conform distribuției q și returnează $f_{GC}(x, q) = sign\left(w^T x\right)$. Spre deosebire de primii doi clasificatori, acesta este o variabilă aleatoare care poate returna etichete diferite la rulări diferite.

2.4 Relații între diferite tipuri de clasificatori

Deoarece în implementare vom folosi primul tip de clasificator, iar teorema din secțiunea 2.5 se referă la cel de-al treilea tip, este necesară prezentarea unei relații între erorile celor două tipuri de clasificatori. Deoarece există și o altă teoremă [5] care oferă o margine superioară pentru clasificatori de tip BVC, bazată pe complexitatea Rademacher, vom demonstra o altă relație între erorile primilor doi clasificatori.

2.4.1 Relația între clasificatorii PC și BVC

Deși, în general, clasificatorii PC și BVC sunt diferiți, următoarea lemă demonstrează că, dacă distribuția folosită de BVC este simetrică în parametrul $\widehat{w}$ folosit de PC, atunci cei doi clasificatori sunt identici.

Lema 1. Fie $\widehat{w}$ parametrul clasificatorului PC. Dacă distribuția q folosită de BVC este simetrică în $\widehat{w}$ (adică $q(w) = q(\widetilde{w})$ unde $\widetilde{w} = 2\widehat{w} - w$), atunci clasa prezisă de PC este aceeași cu clasa prezisă de BVC, pentru orice instanță x.

Demonstrație

Pentru orice w din domeniul integralei, $\widetilde{w} = 2\widehat{w} - w$ este și el în domeniul integralei. Din relația precedentă rezultă $w^T x + \widetilde{w}^T x = 2\widehat{w}^T x$. Există trei cazuri de analizat:

1. $sign\left(w^T x\right) = -sign\left(\widetilde{w}^T x\right)$ sau $w^T x = \widetilde{w}^T x = 0$. În acest caz, contribuția totală a lui w și $\widetilde{w}$ este 0 deoarece

$$sign\left(w^T x\right) q(w) + sign\left(\widetilde{w}^T x\right) q(\widetilde{w}) = 0$$

2. $sign\left(w^T x\right) = sign\left(\widetilde{w}^T x\right)$. Deoarece $w + \widetilde{w} = 2\widehat{w}$, rezultă

$$sign\left(w^T x\right) = sign\left(\widetilde{w}^T x\right) = sign\left(\widehat{w}^T x\right)$$
$$\Rightarrow sign\left(sign\left(w^T x\right) q(w) + sign\left(\widetilde{w}^T x\right) q(\widetilde{w})\right) = sign\left(\widehat{w}^T x\right)$$

3. $w^T x = 0, \widetilde{w}^T x \neq 0$ sau $w^T x \neq 0, \widetilde{w}^T x = 0$. Acest caz este asemănător cazului $2.$, din definiția lui $\widetilde{w}$ rezultând, din nou,

$$sign\left(sign\left(w^T x\right) q(w) + sign\left(\widetilde{w}^T x\right) q(\widetilde{w})\right) = sign\left(\widehat{w}^T x\right)$$

Cu excepția cazului în care $\widehat{w} = 0$, caz în care clasificatorul este nedefinit, cel puțin unul din cazurile 2 sau 3 vor apărea măcar o dată, deci lema este adevărată. Rezultă că erorile celor doi clasificatori considerați (PC și BVC) sunt egale.

2.4.2 Relația între clasificatorii BVC și GC

Lema 2. Eroarea clasificatorului BVC este mai mică sau egală decât dublul erorii clasificatorului GC, dacă se folosește aceeași distribuție.

Demonstrație

Pentru orice instanță etichetată (x, y), presupunem că aceasta este etichetată incorect ca fiind pozitivă de către BVC. Atunci știm că:

$$\int sign\left(w^T x\right) q(w)\, dw > 0$$

Probabilitatea ca GB să clasifice această instanță tot pozitiv este:

$$P\left(f_{GC}=1\right)=\int\left[\left[sign\left(w^{T}x\right)=1\right]\right]\cdot q\left(w\right)dw=\int\frac{1+sign\left(w^{T}x\right)}{2}q\left(w\right)dw$$
$$=\frac{1}{2}+\frac{1}{2}\int sign\left(w^{T}x\right)q\left(w\right)dw>\frac{1}{2}$$

Rezultă că, de fiecare dată când BVC greșește, GC greșește și el cu o probabilitate mai mare decât $\frac{1}{2}$. Deci, eroarea clasificatorului GC este mai mare decât jumătate din eroarea clasificatorului BVC.

Combinând cele două leme, rezultă că, pentru un $\widehat{w}$ fixat, eroarea clasificatorului PC de parametru $\widehat{w}$ este egală cu cea a clasificatorului BVC folosind o distribuție centrată în $\widehat{w}$ și este mai mică sau egală decât dublul erorii clasificatorului GC folosind aceeași distribuție.

2.5 O margine superioară pentru eroarea clasificatorului GC

Teorema, împreună cu demonstrația, este preluată din [5].

Teorema 3. Fie $(x,y)\in\mathbb{R}^{d}\times\{-1,1\}$ o variabilă aleatoare de distribuție D. Fie $S=\{(x_1,y_1),\ldots,(x_m,y_m)\}$ o mulțime de m instanțe de antrenament alese independent conform lui D. Fie $\delta>0$ un număr real. Atunci, cu probabilitate de cel puțin $1-\delta$ peste alegerea lui S, următoarea relație este adevărată:

$$D_{Ber}\left(R_{true}\mid\mid R_{sample}\right)\leq\frac{D\left(P\mid\mid Q\right)+\ln\left(\frac{m+1}{\delta}\right)}{m}$$

unde R_{true} reprezintă eroarea la generalizare a clasificatorului GC folosind distribuția $q(w)$,

$$R_{true}=E_{q(w)}\left[E_{D}\left[l\left(f_{GC}(x,w),y\right)\right]\right],$$

R_{sample} reprezintă eroarea aceluiași clasificator, dar pe mulțimea de antrenament,

$$R_{sample}=E_{q(w)}\left[\frac{1}{m}\sum_{i=1}^{m}l\left(f_{GC}(x_i,w),y_i\right)\right],$$

D_{Ber} a fost definit în 2.2, iar $l(\widehat{y},y)=\mathbf{1}_{\{y\neq\widehat{y}\}}$ reprezintă funcția de pierdere *unu-zero*.

Demonstrație

Fie $p(w,(x,y))=l(f_{GC}(x,w),y)$. Considerăm următoarele funcții:

$$p(w)=E_{(x,y)\sim D}\left[p(w,(x,y))\right],\qquad\widehat{p}(w)=\frac{1}{m}\sum_{i=1}^{m}p(w,(x_i,y_i))$$

Fie $\Delta(w) = D_{Ber}\left[\widehat{p}(w) \mid\mid p(w)\right]$. Fixăm w și notăm $p = p(w)$ și $\widehat{w} = \widehat{p}(w)$. Atunci $m\widehat{p}$ este o variabilă aleatoare peste alegerea lui S și este distribuită binomial de parametri (m, p). Avem:

$$\begin{aligned}
&E_S\left[e^{mD_{Ber}[\widehat{p} \mid\mid p]}\right] \\
&= \sum_{k=0}^{m} \binom{m}{k} \exp\left(m\left(D_{Ber}\left[k/m \mid\mid p\right] + (k/m)\log p + (1-k/m)\log\left(1-p\right)\right)\right) \\
&= \sum_{k=0}^{m} \binom{m}{k} e^{-mH[k/m]}
\end{aligned}$$

unde $H[q] = -q\log q - (1-q)\log(1-q)$ reprezintă entropia unei variabile distribuite Bernoulli de parametru q. Dar $\binom{n}{k} \leq e^{mH[k/m]}$ (demonstrație în [7], example 12.1.3), deci

$$E_S\left[e^{mD_{Ber}[\widehat{p} \mid\mid p]}\right] \leq m+1$$

Făcând media pentru $w \sim P$ și utilizând inegalitatea lui Markov, obținem:

$$P_S\left(E_{w\sim P}\left[e^{m\Delta(w)}\right] > \frac{m+1}{\delta}\right) \leq \delta$$

Fixăm S astfel încât:

$$E_{w\sim P}\left[e^{m\Delta(w)}\right] \leq T, \quad T = \frac{m+1}{\delta}$$

Definim măsura Gibbs

$$dP_G(w) = \frac{e^{m\Delta(w)}}{E_{w\sim P}\left[e^{m\Delta(w)}\right]} dP(w),$$

care este o măsură de probabilitate relativă la $P(w)$. Deoarece entropia relativă a două distribuții este nenegativă, rezultă:

$$\begin{aligned}
0 \leq D\left[Q \mid\mid P_G\right] &= \int \log\left(\frac{E_{w\sim P}\left[e^{m\Delta(w)}\right]}{e^{m\Delta(w)}} \frac{dQ(w)}{dP(w)}\right) dQ(w) \\
&= D\left[Q \mid\mid P\right] + \log E_{w\sim P}\left[e^{m\Delta(w)}\right] - E_{w\sim Q}\left[m\Delta(w)\right].
\end{aligned}$$

Rezultă:

$$E_{w\sim Q}\left[m\Delta(w)\right] \leq D\left[Q \mid\mid P\right] + \log E_{w\sim P}\left[e^{m\Delta(w)}\right]$$

Observând faptul că D_{Ber} este convexă, din inegalitatea lui Jensen rezultă:

$$D_{Ber}\left[E_{w\sim Q}[\widehat{p}(w)] \mid\mid E_{w\sim Q}[p(w)]\right] \leq E_{w\sim Q}\left[D_{Ber}\left[\widehat{p}(w) \mid\mid p(w)\right]\right] \leq \frac{D\left(Q \mid\mid P\right) + \ln\left(\frac{m+1}{\delta}\right)}{m}$$

Din definițiile lui p și $\widehat{p}$, rezultă:

$$R_{true} = E_{w\sim Q}\left[p(w)\right], \quad R_{sample} = E_{w\sim Q}\left[\widehat{p}(w)\right]$$

Înlocuind aceste relații în inegalitatea de mai sus, rezultă:

$$D_{Ber}\left[R_{sample} \mid\mid R_{true}\right] \leq \frac{D\left(Q \mid\mid P\right) + \ln\left(\frac{m+1}{\delta}\right)}{m}$$

2.6 Discuții

Observăm că, spre deosebire de teoremele din capitolul precedent, *teorema 3* nu oferă direct o margine superioară pentru eroarea la generalizare a clasificatorului rezultat. În schimb, oferă o margine superioară pentru $D_{Ber}\left[R_{sample} \mid\mid R_{true}\right]$ care reprezintă divergența *Kullback Leibler (KL-divergence)* a două distribuții Bernoulli de parametri R_{true} și R_{sample}. Astfel, cu cât termenul din partea stângă a inegalității este mai mic, cu atât cele două distribuții sunt mai apropiate, deci R_{true} este apropiat de R_{sample}.

Capitolul 3

Counterfactual Regression

3.1 Introducere

În acest capitol voi introduce un nou tip de probleme care se pot rezolva cu ajutorul unor algoritmi de învățare automată. Pentru a ne familiariza cu unele concepte, voi începe prin a prezenta câteva aplicații practice.

Un exemplu ilustrativ este cel al estimării efectului unui anumit tratament. Considerăm că pentru a vindeca o anumită boală a unei persoane există două tratamente posibile. Doctorul trebuie să decidă care din aceste tratamente va avea un efect mai bun. Un alt exemplu este reprezentat de un profesor care trebuie să decidă ce program de studiu va fi mai benefic pentru un anumit student.

Aceste exemple, precum și acest capitol, se referă la efectul individual: ce efect are o anumită acțiune asupra unui anumit individ. Scopul este acela de a antrena un algoritm pe baza unor observații făcute în trecut. Instanțele de antrenament sunt de forma (x, t, y) care reprezintă faptul că individul x a primit tratamentul t având efectul y. Voi considera o restricție a acestei probleme în cazul în care există exact două acțiuni posibile ($t \in \{0, 1\}$). Instanțele având $t = 0$ vor fi numite instanțe de control, iar cele cu $t = 1$ instanțe tratate. Deoarece sunt două tratamente posibile, există tot două efecte posibile, numite în continuare Y_0 și Y_1 corespunzătoare lui $t = 0$ și $t = 1$.

De exemplu, x poate reprezenta caracteristicile unui pacient cu diabet (factori demografici, rezultatele testelor de laborator, ...), $t = 0$ reprezintă medicamentul clasic de control al glicemiei, iar $t = 1$ reprezintă un nou tratament. $Y_t, t \in \{0, 1\}$, reprezintă nivelul glicemiei dacă pacientului i-ar fi fost administrat tratamentul t [8].

Diferența majoră față de problemele întâlnite până acum este aceea că pentru

fiecare instanță observăm exact un efect, ori Y_0, ori Y_1, niciodată amândouă. O altă problemă care e posibil să apară este aceea a dependințelor ascunse. În cazul exemplului cu pacienții, e posibil ca tratamentul ales să depindă de situația materială. În continuare vom presupune că nu există astfel de dependințe ascunse (pentru exemplul anterior, presupunem că salariul pacientului se regăsește printre atributele din x). Acestă condiție este cunoscută sub numele de *strong ignorability* și este definită formal prin $(Y_0, Y_1) \perp\!\!\!\perp t \mid x$.

Definim două funcții $m_t(x) = E[Y_t \mid x], t \in \{0, 1\}$ reprezentând valoarea medie a efectului tratamentului t asupra instanței x. Efectul tratamentului $t = 1$ relativ la $t = 0$ este funcția $\tau(x) = m_1(x) - m_0(x)$, numită Efectul Tratamentului Individual (*Individual Treatment Effect - ITE*). În general, funcția τ nu este bine definită. O condiție suficientă pentru a depăși această problemă este presupunerea făcută anterior de *strong ignorability*, combinată cu condiția de *overlapping*: $0 < p(t = 1 \mid x) < 1$ [9].

O abordare posibilă de a estima τ este aceea de a învăța cele două funcții m_0 și m_1 din instanțele de antrenament. Însă, dacă distribuțiile $p^{t=0}$ și $p^{t=1}$ sunt foarte diferite, de exemplu majoritatea pacienților cu situație materială bună au primit tratamentul $t = 1$, iar majoritatea pacienților cu situație materială mai puțin bună au primit tratamentul $t = 0$, atunci nu vom avea o estimare prea bună a efectului tratamentului $t = 0$ asupra primei grupe de pacienți.

Pentru a rezolva această problemă, *Shalit et al.* propun următoarea soluție în [8]: învățarea unei ipoteze comune pentru ambele tratamente, folosind o altă reprezentare a instanțelor, care să minimizeze o sumă ponderată dintre funcția de pierdere și distanța dintre distribuțiile instanțelor indusă de reprezentare.

3.2 Notații

Notăm cu $p(x, t)$ distribuția instanțelor peste $X \times \{-1, 1\}$.

Distribuțiile instanțelor tratate și de control sunt: $p^{t=1}(x) = p(x \mid t = 1), p^{t=0}(x) = p(x \mid t = 0)$.

Φ este funcția de reprezentare de la X la R, iar Ψ este inversa ei.

Considerăm $p_\Phi(r, t)$ distribuția indusă de Φ pe $R \times \{-1, 1\}$. Definim similar $p_\Phi^{t=1}(r) = p_\Phi(r \mid t = 1), p_\Phi^{t=0}(r) = p_\Phi(r \mid t = 0)$.

$L : Y \times Y \to \mathbb{R}_+$ reprezintă o funcție de pierdere clasică. Folosind această funcție, definim $l_{h,\Phi}(x, t) = \int_Y L(Y_t, h(\Phi(x), t)) p(Y_t \mid x) dY_t$, valoarea medie a funcției

$h\left(\Phi(x), t\right)$ pentru instanța x și tratamentul t fixate.

Pierderea factuală și contrafactuală (*factual and counterfactual loss*) este notată prin

$$\epsilon_F\left(h, \Phi\right) = \int_{X \times \{0,1\}} l_{h,\Phi}(x,t)\, p(x,t)\, dxdt,$$

respectiv

$$\epsilon_{CF}\left(h, \Phi\right) = \int_{X \times \{0,1\}} l_{h,\Phi}(x,t)\, p(x,1-t)\, dxdt.$$

Pierderea factuală pentru instanțele tratate și de control este:

$$\epsilon_F^{t=1}\left(h, \Phi\right) = \int_X l_{h,\Phi}(x,1)\, p^{t=1}(x)\, dx,$$

și

$$\epsilon_F^{t=0}\left(h, \Phi\right) = \int_X l_{h,\Phi}(x,0)\, p^{t=0}(x)\, dx,$$

Similar se definesc $\epsilon_{CF}^{t=1}$ și $\epsilon_{CF}^{t=0}$.

Notăm prin $u = p(t=1)$. Următoarea relație rezultă imediat: $\epsilon_F\left(h, \Phi\right) = u\epsilon_F^{t=1}\left(h, \Phi\right) + (1-u)\,\epsilon_F^{t=0}\left(h, \Phi\right)$.

Efectul tratamentului asupra instanței x este $\tau(x) = E\left[Y_1 - Y_0 \mid x\right]$. Media erorii în estimarea efectului tratamentului individual este $\epsilon_{PEHE}(f)$.

Efectul empiric al tratamentului asupra instanței x este $\widehat{\tau}(x) = f(x,1) - f(x,0)$.

Precizia în estimarea efectului eterogen (*Precision in Estimation of Heterogeneous Effect* - PEHE, Hill (2011)) este:

$$\epsilon_{PEHE}(f) = \int_X \left(\widehat{\tau}(x) - \tau(x)\right)^2 p(x) dx$$

Acestă noțiune este similară unei funcții de pierdere și va fi mărginită de teorema din secțiunea 3.4.

$IPM_G(p,q)$ reprezintă metrica probabilistă integrală indusă de familia de funcții G între distribuțiile p și q.

3.3 O margine superioară pentru pierderea contrafactuală

Următoarea lemă mărginește pierderea contrafactuală și reprezintă un pas important în demonstrarea teoremei din secțiunea 3.4. Toate noțiunile care apar au fost introduse în 3.2.

Lema 3. Fie $\Phi : X \to R$ o funcție de reprezentare bijectivă, având inversa Ψ și $h : R \times \{-1, 1\} \to Y$ o ipoteză. Considerăm o familie de funcții $G = \{g : R \to Y\}$ astfel

încât să existe o constantă B_Φ cu proprietatea că $\frac{1}{B_\Phi} \cdot l_{h,\Phi}(\Psi(r), t) \in G$. Următoarea inegalitate este adevărată:

$$\epsilon_{CF}(h, \Phi) \leq (1-u)\,\epsilon_F^{t=1}(h, \Phi) + u\epsilon_F^{t=0}(h, \Phi) + B_\Phi \cdot IMP_G\left(p_\Phi^{t=1}, p_\Phi^{t=0}\right)$$

Demonstrație.

$$\begin{aligned}
&\epsilon_{CF}(h, \Phi) - \left[(1-u)\,\epsilon_F^{t=1}(h, \Phi) + u\epsilon_F^{t=0}(h, \Phi)\right] \\
&= \left[(1-u)\,\epsilon_{CF}^{t=1}(h, \Phi) + u\epsilon_{CF}^{t=0}(h, \Phi)\right] - \left[(1-u)\,\epsilon_F^{t=1}(h, \Phi) + u\epsilon_F^{t=0}(h, \Phi)\right] \\
&= (1-u)\left[\epsilon_{CF}^{t=1} - \epsilon_F^{t=1}\right] + u\left[\epsilon_{CF}^{t=0} - \epsilon_F^{t=0}\right] \\
&= (1-u)\int_X l_{h,\Phi}(x, 1)\left(p^{t=0}(x) - p^{t=1}(x)\right)dx + u\int_X l_{h,\Phi}(x, 0)\left(p^{t=1}(x) - p^{t=0}(x)\right)dx \\
&= (1-u)\int_R l_{h,\Phi}(\Psi(r), 1)\left(p_\Phi^{t=0}(x) - p_\Phi^{t=1}(x)\right)dr \\
&+ u\int_R l_{h,\Phi}(\Psi(r), 0)\left(p_\Phi^{t=1}(x) - p_\Phi^{t=0}(x)\right)dr \\
&= B_\Phi \cdot (1-u)\int_R \frac{1}{B_\Phi} l_{h,\Phi}(\Psi(r), 1)\left(p_\Phi^{t=0}(x) - p_\Phi^{t=1}(x)\right)dr \\
&+ B_\Phi \cdot u\int_R \frac{1}{B_\Phi} l_{h,\Phi}(\Psi(r), 0)\left(p_\Phi^{t=1}(x) - p_\Phi^{t=0}(x)\right)dr \\
&\leq B_\Phi \cdot (1-u) \cdot \sup_{g\in G}\left|\int_R g(r)\left(p_\Phi^{t=0}(r) - p_\Phi^{t=1}(r)\right)dr\right| \\
&+ B_\Phi \cdot u \cdot \sup_{g\in G}\left|\int_R g(r)\left(p_\Phi^{t=1}(r) - p_\Phi^{t=0}(r)\right)dr\right| \\
&= B_\Phi \cdot IPM_G\left(p_\Phi^{t=1}, p_\Phi^{t=0}\right)
\end{aligned}$$

3.4 O margine superioară pentru *Precision Estimation of Heterogeneous Effect*

Partea principală a acestui capitol este următoarea teoremă care mărginește valoarea *PEHE* a unei ipoteze f.

Teorema 4. În condițiile Lemei 3, dacă funcția de pierdere L, din definițiile lui ϵ_F și ϵ_{CF}, este pătratică, atunci:

$$\begin{aligned}
\epsilon_{PEHE}(h, \Phi) &\leq 2\left(\epsilon_{CF}(h, \Phi) + \epsilon_F(h, \Phi) - 2\sigma_Y^2\right) \\
&\leq 2\left(\epsilon_F^{t=0}(h, \Phi) + \epsilon_F^{t=1}(h, \Phi) + B_\Phi \cdot IMP_G\left(p_\Phi^{t=1}, p_\Phi^{t=0}\right) - 2\sigma_Y^2\right)
\end{aligned}$$

Demonstrație.

Arătăm prima inegalitate. A doua inegalitate rezultă imediat din **Lema 3**. Reamintim faptul că $\epsilon_{PEHE}(f) = \epsilon_{PEHE}(h, \Phi)$, unde $f(x, t) = h(\Phi(x), t)$.

$$
\begin{aligned}
&\epsilon_{PEHE} \\
&= \int_X \left((f(x,1) - f(x,0)) - (m_1(x) - m_0(x)) \right)^2 p(x) dx \\
&= \int_X \left((f(x,1) - m_1(x)) - (m_0(x) - f(x,0)) \right)^2 p(x) dx \\
&\leq 2 \int_X \left((f(x,1) - m_1(x))^2 + (m_0(x) - f(x,0))^2 \right) p(x) dx \\
&= 2 \int_X (f(x,1) - m_1(x))^2 p(x, t=1) dx + 2 \int_X (m_0(x) - f(x,0))^2 p(x, t=0) dx \\
&+ 2 \int_X (f(x,1) - m_1(x))^2 p(x, t=0) dx + 2 \int_X (m_0(x) - f(x,0))^2 p(x, t=1) dx \\
&= 2 \int_X (f(x,t) - m_t(x))^2 p(x,t) dx dt + 2 \int_X (f(x,t) - m_t(x))^2 p(x, 1-t) dx dt \\
&\leq 2 \left(\epsilon_F - \sigma_Y^2 \right) + 2 \left(\epsilon_{CF} - \sigma_Y^2 \right) \\
&= 2 \left(\epsilon_{CF}(h, \Phi) + \epsilon_F(h, \Phi) - 2\sigma_Y^2 \right)
\end{aligned}
$$

Capitolul 4

Experimente

Pentru a compara acuratețea și eficiența algoritmilor discutați, i-am rulat pe câteva seturi de date reale. Următorul tabel prezintă caracteristicile principale ale seturilor de date utilizate.

Nume	Număr instanțe	Număr atribute	Număr clase	Procent pozitive
Banknote	931	4	2	44%
Satimage	4332	36	7	23%
MNIST	60000	784	10	39%
Letter	13262	16	26	-

Primul set de date, *Banknote Authentication*, are scopul de a construi un clasificator care să diferențieze bancnotele false de cele reale. Conform sursei, *"Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400 × 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images"*.

Cel de-al doilea set de date are scopul de a clasifica diferite texturi pentru care ni se dau la dispoziție imagini făcute din satelit. Conform sursei, *"The database consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values. In the sample database, the class of a pixel is coded as a number"*.

Următorul set de date folosit pentru clasificare este binecunoscutul *MNIST*, utilizat recunoașterea cifrelor scrisă de mână.

Ultimul set de date este utilizat pentru a recunoaște literele alfabetului englez. Conform sursei, *"The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We typically train on the first 16000 items and then use the resulting model to predict the letter category for the remaining 4000. See the article cited above for more details"*.

4.1 Clasificare binară

Am rulat 3 implementări proprii ale algoritmului AdaBoost discutate în capitolul 1: varianta clasică, având funcția de pierdere exponențială, și două generalizări pentru funcții de pierdere logistică și pătratică. Pentru comparație, am rulat atât o variantă de AdaBoost din biblioteca de Python *scikit-learn*, cât și un alt algoritm de boosting numit *XGBoost*.

Deoarece algoritmul AdaBoost este utilizat pentru clasificarea binară, pentru a-l rula pe setul de date *MNIST*, a cărei varibilă de ieșire are 10 clase, am folosit o abordare de tip *1-versus-all*, considerând eticheta 0 ca fiind pozitivă, iar restul fiind toate negative.

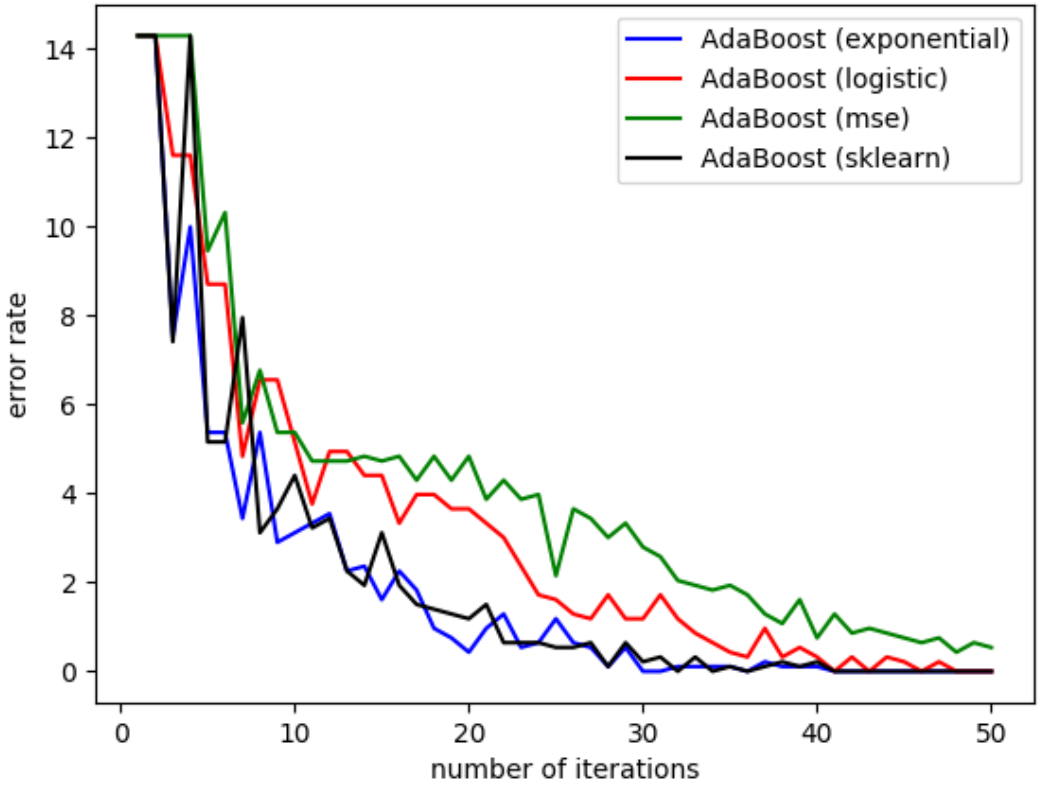

Figura 4.1: Graficul erorii la antrenare pe setul de date *banknote*

Ambele variante ale algoritmului AdaBoost clasic, cât și varianta cu funcția de cost logistică, ajung să aibă eroarea la antrenare 0 după mai puțin de 50 iterații. Pe de altă parte, varianta cu funcție de cost pătratică are nevoie de aproape 350 iterații pentru a ajunge să clasifice corect toate instanțele de antrenament.

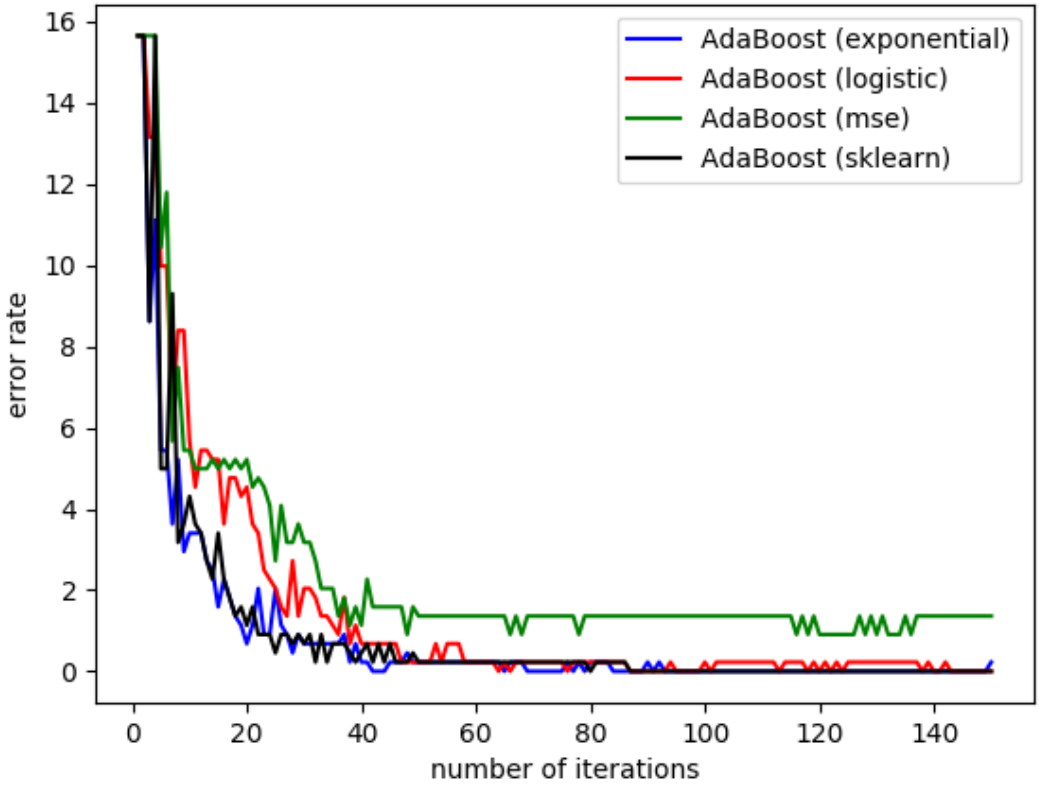

Figura 4.2: Graficul erorii la testare pe setul de date *banknote*

Observăm un fenomen interesant și anume faptul că eroarea la testare continuă să fluctueze chiar și după ce eroarea la antrenare devine 0. Acest lucru poate fi explicat prin faptul că algoritmul nu se oprește din învățat, ci continuă să obțină un clasificator din ce în ce mai bun.

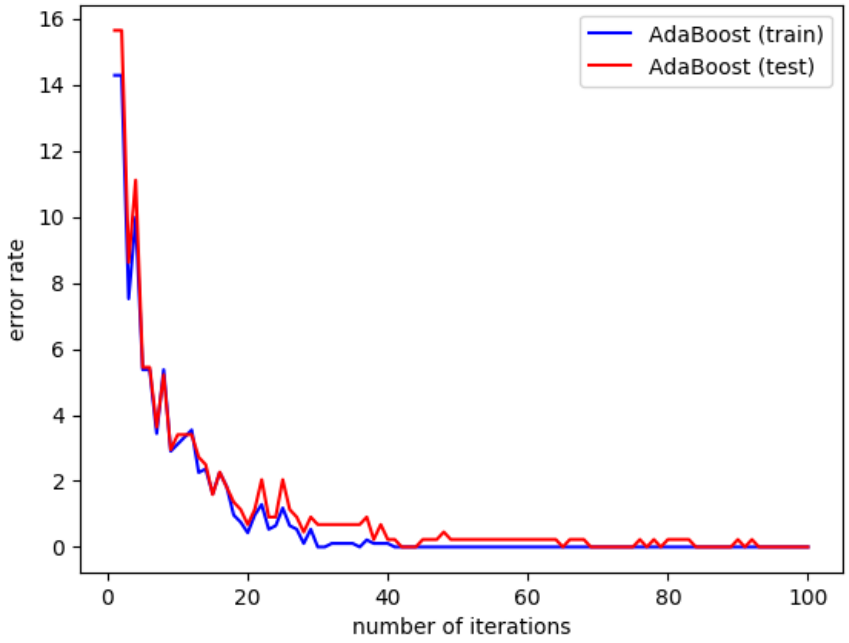

Figura 4.3: Graficul erorii la antrenare, respectiv testare, al algoritmului AdaBoost pe setul de date *banknote*

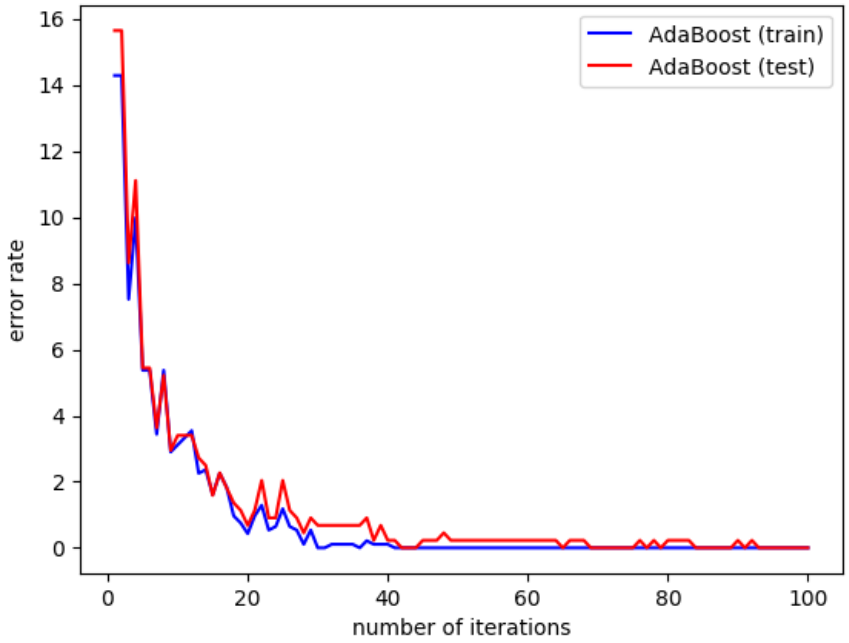

Figura 4.4: Graficul erorii la testare pe setul de date *banknote*

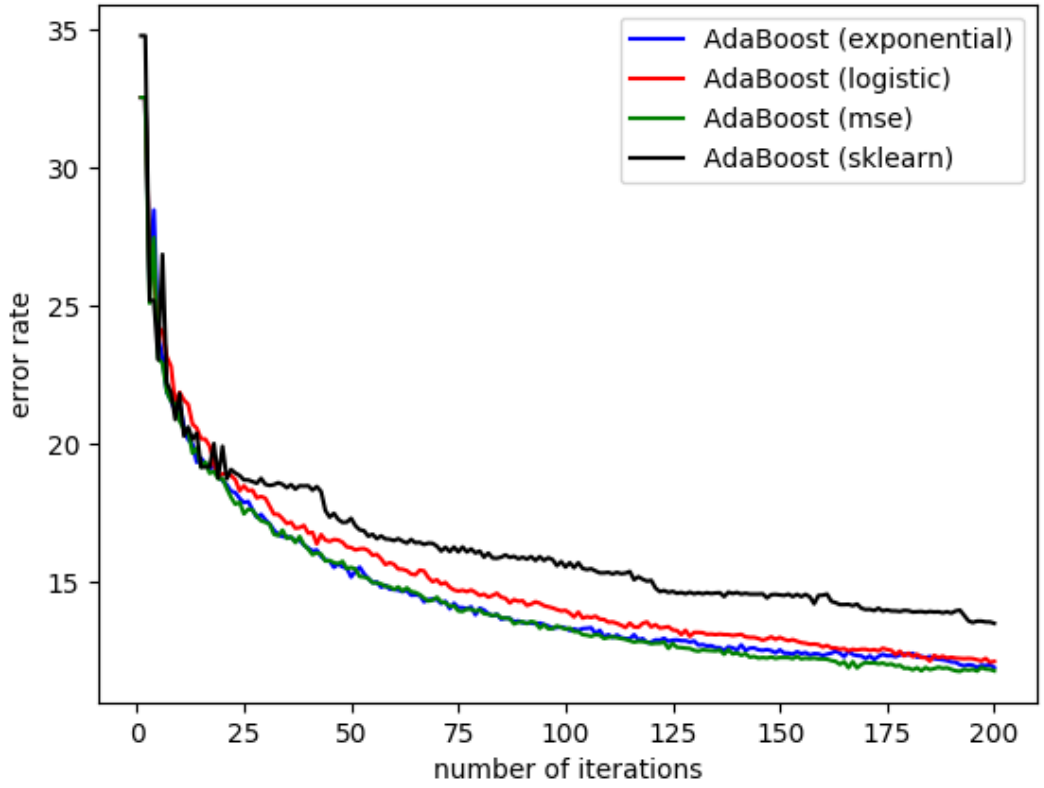

Figura 4.5: Graficul erorii la testare pe setul de date *MNIST*

Pentru a putea rula algoritmul AdaBoost pe acest set de date, am făcut următoarea transformare: cifrele între 0 și 5 au fost considerate instanțe negative, iar restul pozitive.

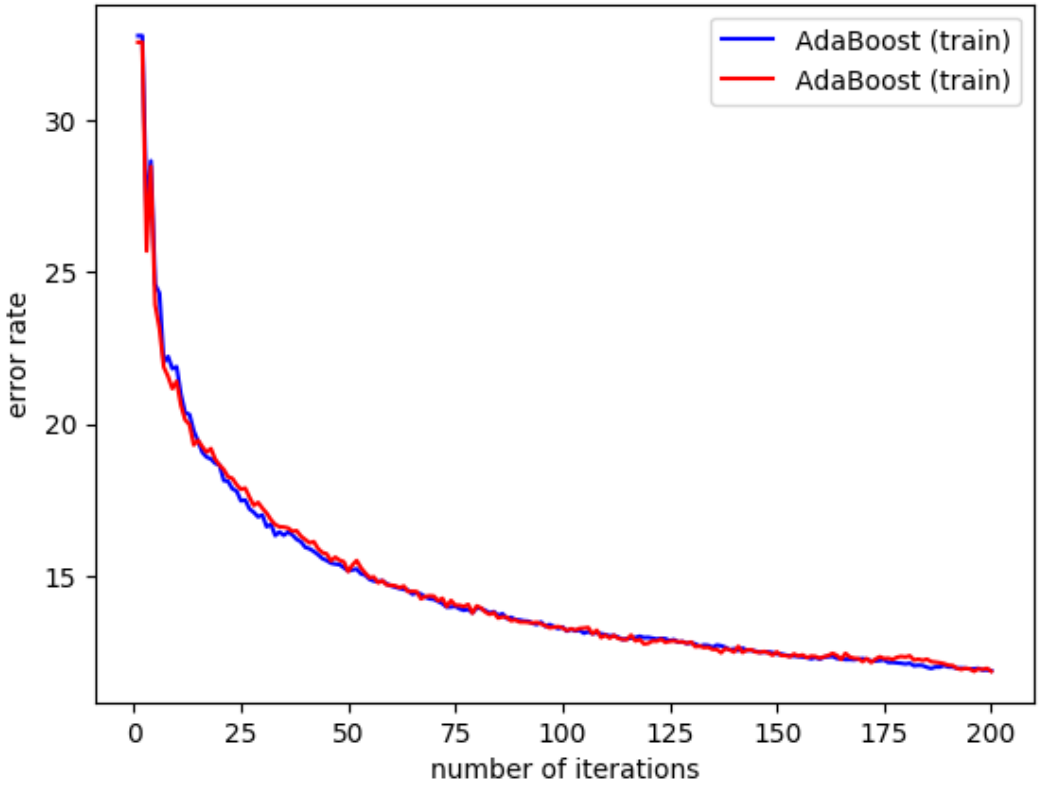

Figura 4.6: Graficul erorii la antrenare, respectiv testare, al algoritmului AdaBoost pe setul de date *MNIST*

4.2 Clasificare *multiclass*

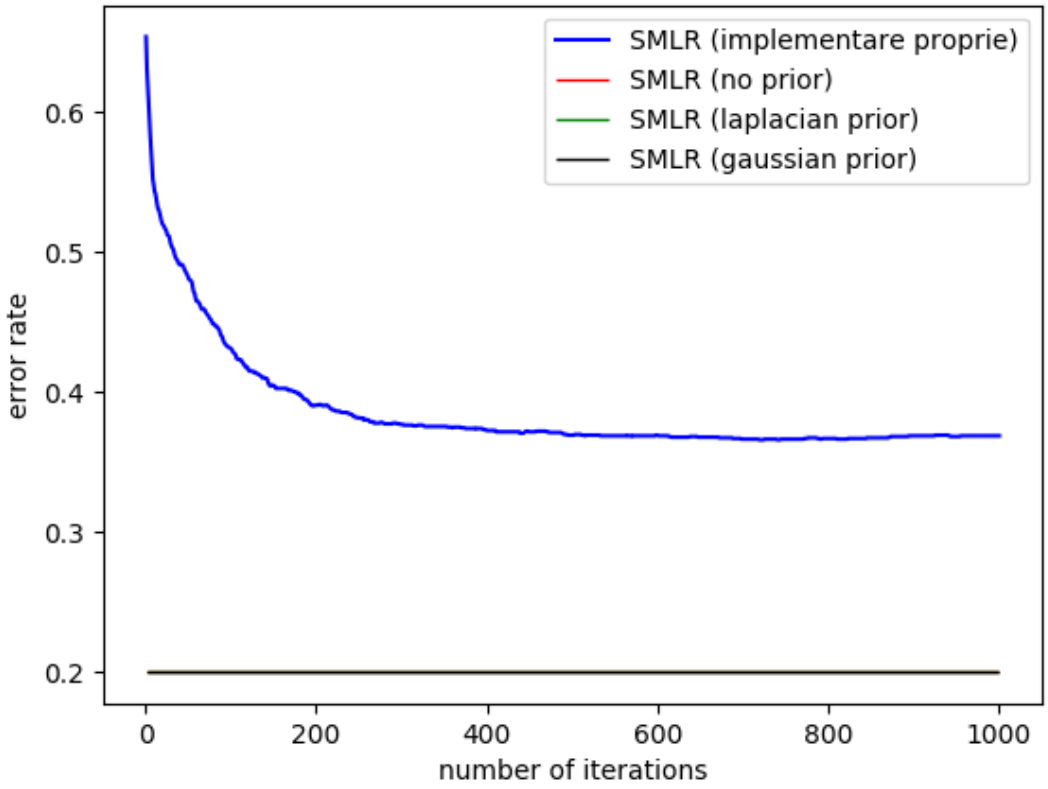

Figura 4.7: Graficul erorii la testare al algoritmului SMLR pe setul de date *satimage*

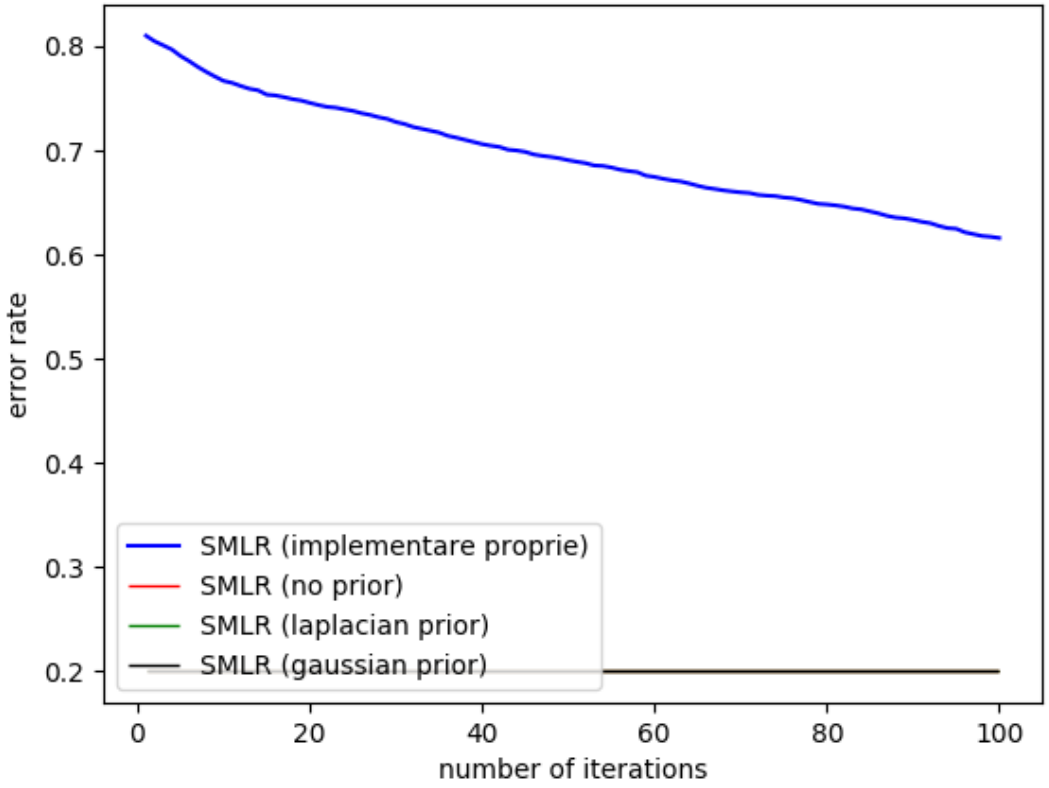

Figura 4.8: Graficul erorii la testare al algoritmului SMLR pe setul de date *letter*

Pentru ambele rulări, parametrul de regularizare λ a fost setat la valoarea 0.1.

Concluzii

Domeniul învățării automate este unul vast, iar algoritmii prezentați în primele două capitole pot fi aplicați pe foarte multe probleme care apar în practică. De aceea, cunoașterea caracteristicilor a diferiți algoritmi poate fi de folos atunci când trebuie să decidem ce metodă vom folosi pentru a rezolva o anumită problemă.

Teoremele prezentate, împreună cu discuțiile făcute pe baza lor, ar trebui să ofere o înțelegere mai profundă a comportamentului unor algoritmilor în practică.

Pe viitor, se pot considera și alte metrici de performanță pentru algoritmi. De exemplu, pentru un set de date cu două variabile de ieșire (clasificare binară), unde 90% din instanțe au aceeași clasă, metrica clasică (numărul de instanțe clasificate corect) nu este cea mai bună (decât dacă acuratețea este foarte mare, cum ar fi $> 99\%$). Acest lucru se datorează faptului că un clasificator extrem de simplu care clasifică toate instanțele la fel ar avea acuratețe de 90%.

De asemenea, deși marginile oferite explică multe lucruri ale comportamentului algoritmilor, în general aceștia se comportă mult mai bine în practică. Totodată, numărul de instanțe necesar pentru ca aceste teoreme să ofere margini relevante este imens.

Observăm că teoremele nu depind de distribuția instanțelor. În practică, uneori, putem avea cunoștințe asupra acestei distribuții și este posibil să existe margini mai bune, care să depindă de aceste distribuții fixate.

Bibliografie

[1] Robert E. Schapire, Yoav Freund, Peter Bartlett și Wee Sun Lee. *Boosting the margin: a new explanation for the effectiveness of voting methods*. The Annals of Statistics, Vol. 26, No. 5, 1651-1686, 1998

[2] Robert E. Schapire și Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012

[3] Maria-Florina Balcan. *Lecture 10: September 22nd, 20118803 Machine Learning Theory*. Georgia Tech, 8803 Machine Learning Theory, Fall 2011

[4] Luc Devroye. *Bounds for the uniform deviation of empirical measures*. Jornal of multivariate analysis, 12, 72-79, 1982

[5] Balaji Krishnapuram, Lawrence Carin, Mário A.T. Figueiredo și Alexander J. Hartemink. *Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds*. IEEE transactions on pattern analysis and machine intelligence, Vol. 27, No. 6, 2005

[6] Matthias Seeger. *PAC-Bayesian Generalization Error Bounds for Gaussian Process Classification*. Journal of Machine Learning Research, Vol. 3, 233-269, 2002

[7] Thomas M. Cover, Joy A. Thomas. *Elements of Information Theory*. Series in Telecommunications. John Wiley & Sons, 1st edition, 1991

[8] Uri Shalit, Fredrik D. Johansson, and David Sontag. *Estimating individual treatment effect: generalization bounds and algorithms*. ArXiv e-prints, 2017

[9] Guido M. Imbens and Jeffrey M. Wooldridge. *Recent Developments in the Econometrics of Program Evaluation*. Journal of Economic Literature 47, no. 1, 5-86, 2009

[10] Adith Swaminathan, Thorsten Joachims. *Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization*. Journal of Machine Learning Research 16, 1731-1755, 2015