UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Clasificare versus filtrare colaborativă în predicția clinică

propusă de

Mălina Cătonoiu

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Liviu Ciortuz

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Clasificare versus filtrare colaborativă în predicția clinică

Mălina Cătonoiu

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Liviu Ciortuz

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Liviu Ciortuz.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnata **Cătonoiu Mălina** domiciliată în **România, jud. Neamț, mun. Roman, str. Martir Cloșca, nr. 23**, născută la data de **19 iunie 1997**, identificat prin CNP **2970619270836**, absolventă a Facultății de informatică, **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 **Clasificare versus filtrare colaborativă în predicția clinică** 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.

Data: ..... Semnătura:

.....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Clasificare versus filtrare colaborativă în predicția clinică**, 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 **Mălina Cătonoiu**

Data: ..... Semnătura:

.....

Cuprins

Introducere

Odată cu creșterea cantității de date disponibile și intrarea în era *Big Data*, explorarea unor noi metode de învățare devine necesară. Astfel, în ultima perioadă, sistemele de recomandare, în special filtrarea colaborativă, au câștigat popularitate, aplicațiile lor fiind extinse către domenii netradiționale, cum ar fi diagnoza medicală.

În această lucrare vom încerca definirea algoritmilor de clasificare și a celor de filtrare colaborativă, și apoi aplicarea lor pe probleme din domeniul medical.

Capitolul I cuprinde caracterizarea algoritmilor de clasificare. De asemenea, se va pune în discuție estimarea parametrilor acestora, necesară în optimizare.

Capitolul II descrie filtrarea colaborativă și doi algoritmi care pot fi aplicați cu această abordare: k-Nearest Neighbors și Single Value Decomposition. Vom efectua și un exemplu aplicativ, pentru a observa cum se comportă acești algoritmi atunci când sunt aplicați pe o problemă clasică de filtrare colaborativă.

Capitolul III prezintă comparația dintre cele două clase de algoritmi, efectuată pe mulțimi de date din domeniul medical. Vom vedea cum se comportă filtrarea colaborativă atunci când este aplicată pe probleme clasice de clasificare.

Capitolul 1

Clasificarea

1.1 Definirea algoritmilor de clasificare

Învățarea supervizată este problema de a învăța o funcție care pune în corelație o dată de intrare cu una de ieșire, pe baza unor exemple de perechi de date intrare - ieșire.

Vom stabili o notație pe care să o folosim ulterior, unde $x^{(i)}$ vor fi variabilele de intrare, numite de asemenea și atribute, iar $y^{(i)}$ va fi variabila de ieșire sau variabila target pe care încercăm să o prezicem. Perechea $(x^{(i)}, y^{(i)})$ se va numi instanță de antrenament, iar setul de date folosit la învățare - o listă de m instanțe de antrenare $(x^{(i)}, y^{(i)}); i = 1, ..., m$ - se numește set de antrenament. Vom nota de asemenea cu $\mathcal{X}$ spațiul variabilelor de intrare și cu $\mathcal{Y}$ spațiul variabilelor de ieșire.

Atunci când variabila target y ia un număr mic de valori discrete, numim problema de învățare clasificare.

Pentru a descrie în mod formal problema învățării supervizate, scopul nostru este ca, având un set de antrenament, să învățăm o funcție $h : \mathcal{X} \to \mathcal{Y}$, astfel încât $h(x)$ să fie un predictor bun pentru valoarea lui y. Funcția h se numește ipoteză.

Exemple clasice de probleme în care se folosește clasificarea sunt: identificarea email-urilor spam sau diagnosticarea unui pacient pe baza caracteristicilor acestuia (gen, tensiunea arterială, prezența/absența anumitor simptome, etc.)

1.2 Regresia Logistică

Regresia Logistică este un algoritm de clasificare, care lucrează urmărind să învețe o funcție care să aproximeze în mod convenabil distribuția $P(Y|X)$. Presupunerea ei de

bază este că $P(Y|X)$ poate fi aproximată cu o funcție sigmoidală (logistică), aplicată unei combinații liniare de atribute de intrare. În cazul regresiei logistice, $y \in 0, 1$

Pentru a contura cât mai riguros algoritmul, vom începe prin a defini funcția care transformă datele de intrare într-o predicție pentru instanța dată, pe care o vom numi în continuare ipoteza.

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1.1}$$

unde

$$g(z) = \frac{1}{1 + e^{-z}} \tag{1.2}$$

se numește funcție logistică sau sigmoidă.

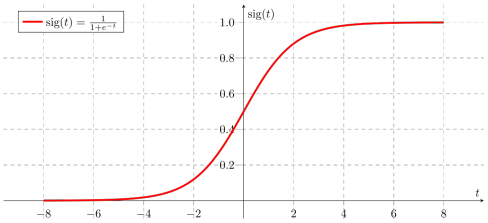

Figura 1.1: *Graficul funcției sigmoide.*

Funcția sigmoidă are rolul de a mapa orice număr real pe intervalul (0, 1), pentru că, așa cum se poate observa în figura alăturată, pe măsură ce $z \to \infty$ $g(z)$ tinde la 1, iar când $z \to -\infty$ $g(z)$ tinde la 0. Această proprietate este folositoare pentru a transforma o funcție cu valori arbitrare într-o funcție mai potrivită pentru clasificare.

Ipoteza $h_\theta(x)$ exprimă probabilitatea ca outputul să fie 1. Ca exemplu, dacă avem $h_\theta(x) = 1$, înseamnă că probabilitatea ca outputul nostru să fie egal cu 1 este de 70%. Așadar, putem rescrie ipoteza ca:

$$h_\theta(x) = P(y = 1|x; \theta) \tag{1.3}$$

Ținând cont de faptul că probabilitatea totală trebuie să însumeze 1, vom avea:

$$1 - h_\theta(x) = P(y = 0|x; \theta) \tag{1.4}$$

Pentru a măsura abilitatea modelului de a estima relația dintre X și y, algoritmul folosește o funcție de cost, definită astfel:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{1.5}$$

unde

$$\begin{cases} Cost(h_\theta(x), y) = -\log(h_\theta(x)) & \text{dacă y = 1} \\ Cost(h_\theta(x), y) = -\log(1 - h_\theta(x)) & \text{dacă y = 0} \end{cases}$$

După cum se poate observa din grafic, atunci când y = 1 și $h_\theta(x) = 1$, funcția cost tinde la 0. Pe de altă parte, când y = 1 și $h_\theta(x) = 1$, funcția tinde la infinit. Analog

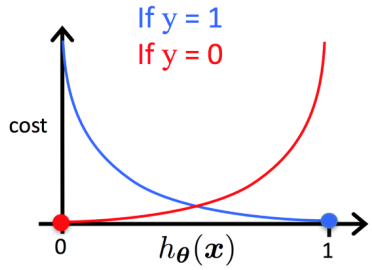

Figura 1.2: Graficul funcției cost pentru y = 0 și y = 1.

Sursa: https://datascience.stackexchange.com/questions/40982/logistic-regression-cost-function

pentru cel de al doilea caz. Aceasta este o proprietate importantă a funcției cost: pe măsură ce diferența dintre y și $h_\theta(x)$ crește, va crește și penalizarea aplicată algoritmului.

Putem rescrie cele două condiții ale funcției cost în una singură astfel:

$$Cost(h_\theta(x), y) = -\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)) \tag{1.6}$$

Atunci când y = 1, al doilea termen $(1-y)\log(1-h_\theta(x))$ va fi 0, așadar nu afectează rezultatul. De asemenea, când y = 0, primul termen al ecuației va fi 0.

Vom putea rescrie acum funcția de cost în felul următor:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x)) + (1-y^{(i)})\log(1-h_\theta(x))] \tag{1.7}$$

Estimarea parametrilor

Având această funcție cost, ceea ce încearcă algoritmul să realizeze este să găsească parametrii θ astfel încât să minimizeze $J(\theta)$. Aceasta se realizează folosind metoda gradientului descendent. Ideea acestui algoritm este că atunci când facem în mod continuu pași mici în direcția gradientului funcției de optimizat, gradientul fiind vectorul de derivate parțiale ale funcției respective, vom ajunge la un moment dat într-un punct de minim local al acelei funcții.

Forma generală a gradientului descendent este:

$$\theta_j^{new} = \theta_j^{old} - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \tag{1.8}$$

unde α este un factor constant care determină mărimea pasului executat la fiecare iterație, cunoscut sub numele de *rată de învățare*.

Derivata parțială a funcției $J(\theta)$ are următoarea formă:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x^{(i)} - y^{(i)}) x_j^{(i)}) \tag{1.9}$$

Vom rescrie acum pasul gradientului descendent:

$$\theta_j^{new} = \theta_j^{old} - \alpha \sum_{i=1}^{n} (h_\theta(x^{(i)} - y^{(i)}) x_j^{(i)}) \tag{1.10}$$

Regularizarea Regresiei Logistice

Fenomenul de *overfitting* se produce atunci când funcția ipoteză învață foarte bine datele curente, dar are putere mică de generalizare. Pentru a rezolva această problemă există două abordări:

1. Reducerea numărului de trăsături - realizabilă prin selectarea manuală a trăsăturilor care vor fi folosite sau prin folosirea unui algoritm de alegere a modelului

2. Regularizare - păstrarea tuturor atributelor, dar reducerea magnitudinii parametrilor θ_j

   Pentru a aplica regularizarea vom rescrie pașii pentru gradientul descendent astfel:

$$\begin{cases} \theta_0^{new} = \theta_0^{old} - \alpha \frac{1}{m} \sum_{i=1}^{n} (h_\theta(x^{(i)} - y^{(i)}) x_0^{(i)}) \\ \theta_j^{new} = \theta_j^{old} - \alpha [\frac{1}{m} \sum_{i=1}^{n} (h_\theta(x^{(i)} - y^{(i)}) x_0^{(i)}) + \frac{\lambda}{n} \theta_j] \end{cases}$$

În funcție de modul în care este penalizat parametrul θ există mai multe tipuri de regularizări, dintre care le vom aminti pe cele mai importante două:

1. Regularizarea L1 - modelul care folosește această regularizare se va numi Regresie Lasso. Pentru această regresie, termenul de regularizare va fi $\lambda \sum_{i=0}^{d} |\theta_j|$.

2. Regularizarea L2 - modelul care folosește această regularizare se va numi Regresie Ridge. Acest tip de regresie penalizează pătratul mărimii vectorului θ. Termenul de regularizare este $\lambda \sum_{i=0}^{d} \theta_j^2$.

Pentru că în experimentul pe care îl vom efectua în Capitolul III avem și seturi de date nominale, vom continua prin a descrie cum poate fi generalizat algoritmul Regresie Logistică.

1.2.1 Regresia Logistică Multinomială

Regresia Logistică multinomială este o generalizare a regresiei logistice pentru problemele *multiclass*. Este folosită atunci când variabila dependentă Y este nominală (poate lua mai mult de două valori discrete).

Pentru a folosi algoritmul pe acest gen de probleme, vom aborda perspectiva *One-vs-All*.

Astfel, pentru o problemă de clasificare unde $y \in 0, 1, ..., n$, vom avea n + 1 cazuri de clasificare binară. Pentru cazul i, $1 \leq i \leq n$, problema se rezumă la a decide dacă o instanță face parte din clasa i. Așadar, clasificatorul are următoarele opțiuni:

$$\begin{cases} 1, \text{ dacă y = i} \\ 0, \text{ altfel} \end{cases}$$

1.3 Support Vector Machine

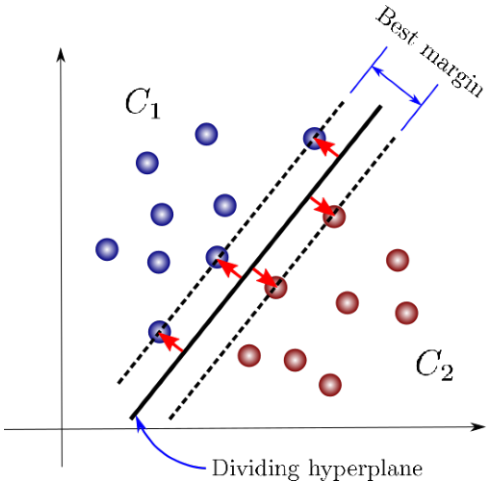

Figura 1.3: *Ilustrarea clasificării conform SVM.*

Sursa: https:// towardsdatascience.com

Support Vector Machine este un algoritm de clasificare binară care urmărește să găsească hiperplanul care separă în mod optim cele două clase. În spațiul bidimensional, acest hiperplan este o dreaptă care separaă un plan în două părți, corespunzătoare celor două clase, după cum se poate observa în figura alăturată.

Ideea de bază a acestui algoritm este că modelul care generalizează cel mai bine este acela care desparte cel mai mult clasele, adică cel care are marginea cea mai mare între clase. Marginea este distanța formată între cele mai apropiate puncte care aparțin unor clase distincte și hiperplanul separator.

Pentru clasificare, vom folosi o funcție liniară, de forma

$$f(x) = w \cdot x, \text{unde } w \in \mathbb{R}^d \tag{1.11}$$

O instanță oarecare x va fi clasificată în clasa 1 dacă $f(x)0$, respectiv în clasa -1 în caz contrar. Hiperplanul de ecuație $f(x) = 0$ va funcționa ca separator al instanțelor de antrenamen, sau graniță de decizie.

1.3.1 Kernelizarea

În practică, există multe situații în care datele nu sunt separabile liniar. Pentru a putea fi aplicată separarea liniară, se vor transforma datele din spațiul inițial al problemei, numit spațiul atributelor, într-un spațiu cu mult mai multe dimensiuni, numit spațiul trăsăturilor. În acest spațiu, datele devin separabile liniar. Această transformare se face cu ajutorul unei funcții.

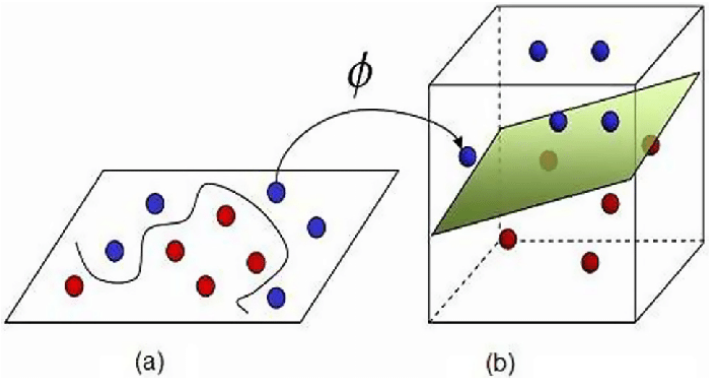

Figura 1.4: Ilustrarea kernelizării.

1.4 Concluzie capitol

În acest capitol am descris algoritmii de clasificare și am menționat principalii parametri ai acestora, care pot fi folosiți pentru optimizare.

Capitolul 2

Filtrarea Colaborativă

Un sistem de recomandare este o subclasă a sistemelor de filtrare a informației, care urmărește să prezică preferința unui utilizator pentru un obiect. Domeniul principal în care se aplică această abordare este cel comercial.

Evenimentul care a impulsionat cercetarea în această arie de studiu a fost competiția Netflix.

Pentru a recomanda conținut suplimentar, sistemele de recomandare moderne se folosesc de feedback-ul utilizatorilor din trecut. Există două modalități complementare de a formaliza această problemă: recomandările bazate pe conținut și filtrarea colaborativă[5].

Pentru recomandările bazate pe conținut, un obiect este reprezentat ca un vector de trăsături, la fel ca în problemele clasice de clasificare sau regresie.

În schimb, în filtrarea colaborativă, un obiect este reprezentat în funcție de felul în care a fost notat de alți utilizatori, renunțându-se complet la observarea oricăror trăsături explicite despre acesta.

2.1 Definiția filtrării colaborative

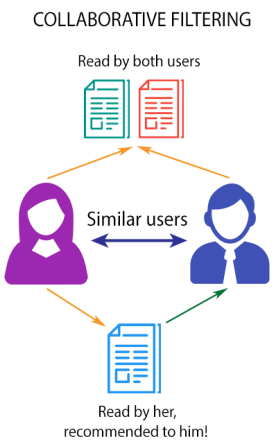

Figura 2.1: *Exemplu ilus-*

Filtrarea colaborativă se bazează pe presupunerea că cei care au avut interese comune vor avea și în viitor, și că vor continua să aprecieze tipul de obiecte pe care îl apreciau și în trecut. Astfel, recomandările sunt genereate folosindu-se doar informații despre profilul de notare al diferiților utilizatori sauu obiecte. Găsind utilizatori cu un istoric de notare similar cu cel al utilizatorului curent, pot fi generate recomandări bazate pe această vecinătate. Avantajul major al

acestei abordări constă în faptul că nu se bazează doar pe conținutul ușor analizabil, astfel încât este capabilă să facă recomandări pertinente pentru obiecte complexe, cum ar fi filmele, fără să necesite o înțelegere a obiectului în sine.

Țelul unui algoritm de filtrare colaborativă este de a sugera noi obiecte sau de a prezice utilitatea unui anumit obiect pentru un anumit utilizator, bazându-se pe preferințele anterioare ale utilizatorului și pe opiniile altor utilizatori asemănători. Așadar, un scenariu tipic de filtrare colaborativă constă într-o listă de m utilizatori $\mathcal{U} = u_1, u_2, ..., u_m$ și o listă de n obiecte $\mathcal{I} = i_1, i_2, ..., i_n$. Fiecare user u_i are o listă de obiecte I_{u_i} asupra cărora și-a exprimat opinia. Opiniile pot fi date explicit de către utilizator ca o notă, în mod normal aparținând unui anumit interval numeric, sau pot fi extrase implicit din istoricul tranzacțiilor, minând hyperlink-uri web, etc. Există un utilizator particular, pe care îl vom nota $u_a \in \mathcal{U}$, numit *utilizatorul curent*, pentru care sarcina algoritmului de filtrare colaborativă de a găsi obiecte corespunzătoare poate avea două forme:

- Predicția este o valoare numerică $P_{a,j}$ care exprimă nota prezisă a obiectului $i_j \notin I_{u_a}$ pentru utilizatorul curent u_a. Această valoare este cuprinsă în același interval precum celelalte opinii ale utilizatorului u_a.

- Recomandarea este o listă de N obiecte, $I_r \subset \mathcal{I}$, pe care utilizatorul curent le va aprecia cel mai mult. Aceste obiecte trebuie să nu facă parte din lista celor deja evaluate, adică $I_r \cap I_{u_a} = \emptyset$

Filtrarea colaborativă poate fi, la rândul ei, împărțită în două subcategorii: filtrarea bazată pe memorie și cea bazată pe model. În continuare, vom caracteriza aceste metode, iar în final le vom aplica pe o problema clasică de filtrare colaborativă, pentru a vedea cum funcționează în practică.

2.1.1 Filtrare Item-Based

Filtrarea bazată pe conținut a fost introdusă de Amazon în 1996. Până atunci erau folosite doar sisteme de recomandare bazate pe asemănarea dintre utilizatori. Această abordare presupune calcularea similarității între diferite obiecte din setul de date, folosind una dintre metricile de similaritate.

Gradul de asemănare dintre două obiecte este măsurat observând toți utilizatorii care au notat ambele obiecte.

2.1.2 Filtrare User-Based

Filtrarea colaborativă bazată pe utilizatori caută utilizatorii care sunt cei mai similari cu utilizatorul curent. După ce aceștia au fost găsiți, utilizatorul curent va primi recomandări formate din obiectele care au primit note mari din partea celorlalți, dar care încă nu au fost notate de el.

2.2 Filtrare bazată pe memorie

Abordarea bazată pe memorie folosește toată baza de date pentru a rezolva problema prezicerii unui rating. Se încearcă găsirea utilizatorilor cei mai similari cu utilizatorul curent, iar apoi preferințele acestora sunt folosite pentru a face preziceri pentru utilizatorul curent.

2.2.1 k-Nearest Neighbors

Algoritmul k-Nearest Neighbors este o metodă non-parametrică folosită în clasificare și regresie. În ambele cazuri, inputul constă în cele mai apropiate k exemple de antrenament din spațiul de trăsături. k-NN este un tip de învățare bazat pe instanțe, sau lazy learning, în care funcția este aproximată local și toate calculele sunt amânate până în momentul clasificării.

Exemplele de antrenament sunt vectori într-un spațiu multidimensional de trăsături, fiecare având eticheta clasei din care aparțin. În faza de antrenament algoritmul doar stochează acești vectori împreuna cu etichetele corespunzătoare. În faza de clasificare, unui vector neetichetat îi va fi asignată o clasă în funcție de cele mai apropiate k exemple de antrenament. k este o constantă definită de către utilizator.

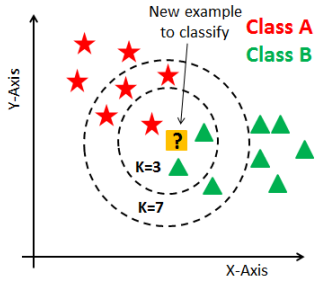

Figura 2.2: *Exemplu de clasificare folosind k-NN, pentru $k = 3$ și $k = 7$.*

Pentru a se determina distanța dintre instanțe cea mai folosită metrică este distanța Euclidiană. Astfel, presupunem o instanță oarecare x, descrisă de vectorul de trăsături $a_1(x), a_2(x), ..., a_n(x)$, unde $a_r(x)$ exprimă valoarea celui de al r-lea atribut al lui x. Atunci,

distanța euclidiană dintre două instanțe x_i și $x_j$va fi:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

Pentru a aplica k-NN în contextul filtrării colaborative vom avea nevoie de metode de măsurare a similarității dintre instanțe. Cele mai utilizate două astfel de metrici sunt coeficientul de corelație Pearson și similaritatea cosinului.

Coeficientul Pearson este o măsură pentru determinarea corelației dintre două variabile X și Y. Adaptat pentru sisteme de recomandare, acesta măsoară cât de mult variază doi utilizatori, împreună, de la voturile lor medii - așadar, direcția votului fiecăruia în comparație cu media voturilor lor. Dacă variază în aceeași direcție, atunci vor fi pozitiv corelați; altfel, vor fi negativ corelați.

Coeficientul de corelație Pearson are următoarea formula:

$$\frac{\sum_{i \in I_{uv}} ((r_{ui} - \bar{u})(r_{vi} - \bar{v}))}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{v})^2}}, unde:$$

- I_{uv} este setul tuturor filmelor notate de ambii utilizatori
- r_{ui} ratingul oferit de utilizatorul u filmului i, analog pentru r_{vi}
- $\bar{u}$ media tuturor ratingurilor oferite de către utilizatorul u, analog pentru $\bar{v}$

Similaritatea cosinului tratează doi utilizatori ca vectori într-un spațiu n-dimensional, unde n este numărul de obiecte din setul de date, și măsoară cosinul unghiului dintre aceștia. Doi vectori cu aceeași orientare vor avea similaritatea cosinului egală cu 1, iar doi vectori diametral opuși o vor avea egală cu -1.

Formula pentru similaritatea cosinului este:

$$\frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}, unde:$$

- I_{uv} este setul tuturor filmelor notate de ambii utilizatori
- r_{ui} ratingul oferit de utilizatorul u filmului i, analog pentru r_{vi}

Abordarea k-NN în filtrarea colaborativă are multe beneficii. Este simplă din punct de vedere conceptual, ușor de implemententat, de obicei nu presupune mulți parametri care trebuie ajustați. Funcționează cel mai bine pentru utilizatorii care au preferințe stereotipice - astfel utilizatorul curent poate fi foarte ușor încadrat într-un grup, însă metoda nu mai e la fel de potrivită atunci când există utilizatori cu interese mixte față de ceilalți.

2.3 Filtrare bazată pe model

2.3.1 Single Value Decomposition

Descompunerea valorilor singulare este un algoritm de factorizare de rang jos a matricilor reale sau complexe, într-o matrice optimală din punct de vedere al erorii pătrate. Vom introduce descompunerea valorilor singulare ale unei matrice prin următoarea teoremă.

Orice matrice $A \in C^{m \times n}$ are o factorizare de forma $U\Sigma V^*$, unde:

- U este o matrice unitară, de dimensiuni $m \times m$, (ortogonală, când $C = R$)
- Σ este o matrice diagonală de dimensiuni $m \times n$, care conține numere reale nonnegative pe diagonală
- V este o matrice unitate de dimensiuni $n \times n$, (ortogonală, când $C = R$)
- V^* este conjugata transpusă a lui V

Valorile de pe diagonala principală din matricea Σ se numesc valorile singulare ale matricei A.

2.4 Exemplu aplicativ

Pentru a vedea cum funcționează filtrarea colaborativă, vom aplica cei doi algoritmi menționați anterior pe o problemă clasică din aria sistemelor de recomandare: prezicerea notei pe care un utilizator o va acorda unui film pe care încă nu l-a vizionat.

Vom folosi Surprise, o librărie Python axată pe sisteme de recomandare. În interiorul acestui pachet avem puse la dispoziție un număr de seturi de date, dintre care îl vom folosi pe cel oferit de către MovieLens. Acesta conține 100.000 de instanțe, un număr relativ mic pentru o problemă de filtrare colaborativă.

Ca măsură de a cuantifica acuratețea algoritmilor vom folosi RMSE - *Root Mean Squared Error*, care se calculează în felul următor:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y_i} - y_i)^2}{n}}, unde:$$

- $\hat{y_1}, \hat{y_2}, ..., \hat{y_n}$ sunt valorile prezise,
- $y_1, y_2, ..., y_n$ sunt valorile observate,

- n este numărul de observații.

Testarea modelelor se va realiza prin cross-validare pe 4 niveluri: [3, 5, 7, 10]. Pentru o aproximare mai fidelă a acurateței, vom rula fiecare algoritm de 20 de ori, iar apoi vom face media rezultatelor.

Algoritmii de filtrare colaborativă se bazează pe găsirea de similitudini între utilizatori, astfel încât numărul de observații are un impact foarte mare asupra acurateței rezultatelor. Setul de date pe care l-am folosit ne oferă un număr rezonabil de astfel de observații, însă, în Capitolul 4 vom aplica acești algoritmi pe seturi de date mult mai mici. Așadar, pentru a observa cum afectează scăderea numărului de instanțe performanța algoritmilor, îi vom rula și pe seturi de date de 1000, respectiv 100 de observații, extrase la întamplare din setul de date complet. Testarea se va realiza așa cum am descris-o anterior.

	k-NN				SVD			
nr. instanțe	3-fold	5-fold	7-fold	10-fold	3-fold	5-fold	7-fold	10-fold
100k	1.021	1.016	1.015	1.014	0.945	0.935	0.932	0.929
1000	1.122	1.1225	1.1221	1.121	1.127	1.150	1.120	1.090
100	1.125	1.101	1.126	1.114	1.132	1.134	1.120	1.081

Deși rezultatele rulării pe întregul set de date sunt mulțumitoare, ținând cont de faptul că nu s-a aplicat nicio metodă de optimizare a modelelor, se poate observa ca acestea se deteriorează pe măsură ce numărul de instanțe scade. Este o observație care trebuie menționată, având în vedere că în capitolul următor vom efectua experimentul pe seturi de date de dimensiuni mici.

În continuare, vom încerca îmbunătățirea performanței prin reglarea hyper-parametrilor. Pentru aceasta vom folosi *Grid Search*, o metodă care determină valorile optimale ale hyper-parametrilor unui algoritm prin căutare exhaustivă.

Hyper-parametrii pe care vom aplica Grid Search vor fi următorii:

1. pentru k-NN:

    - k - numărul maxim de vecini care vor fi luați în considerare. Vom încerca să găsim cea mai bună valoare pentru k din intervalul [10, 50].
    - metrica de similitudine - Vom testa performanța algoritmului în funcție de metrica aleasă (coeficientul Pearson sau similaritatea cosinului).

2. pentru SVD:

   - numărul de iterații ale procedurii SGD (Stochastic Gradient Descent) - vom testa intervalul [1, 40].
   - rata de învățare - vom alege intervalul [0.005, 0.1].
   - termenul de regularizare - vom alege intervalul [0.01, 0.1].

Pentru a putea testa acuratețea algoritmilor optimizați, vom aplica GridSearch pe date pentru a determina valorile optime ale parametrilor, iar apoi vom reface experimentul de mai sus. În urma acestui experiment, am obținut următoarele valori optime:

1. pentru k-NN:

   - k = 50
   - metrica de similitudine = coeficientul Pearson

2. pentru SVD:

   - numărul de iterații ale procedurii SGD (Stochastic Gradient Descent) = 40
   - rata de învățare = 0.005
   - termenul de regularizare = 0.1

	k-NN optimizat				SVD optimizat			
nr. instanțe	3-fold	5-fold	7-fold	10-fold	3-fold	5-fold	7-fold	10-fold
100k	1.010	0.9996	0.995	0.992	0.927	0.919	0.916	0.914
1000	1.121	1.128	1.119	1.116	1.098	1.087	1.088	1.080
100	1.105	1.117	1.106	1.102	1.117	1.126	1.116	1.094

Se poate observa că, în general, performanța algoritmilor a crescut după reglarea hyper-parametrilor.

2.5 Concluzie capitol

În acest capitol am prezentat noțiunile de sisteme de recomandare, filtrare colaborativă, precum și doi algoritmi care se folosesc în această abordare: k-Nearest Neighbour și Single Value Decomposition. În încheiere am aplicat acești algoritmi pe o mulțime de date oferită de MovieLens și am observat cum este afectată performanța lor de cantitatea de date.

Capitolul 3

Comparație între cele două clase de algoritmi

3.1 Motivație

Sistemele de recomandare s-au dovedit a fi foarte eficiente în predicția recomandărilor personalizate utilizatorilor în diferite domenii (marketing, comerț online, etc). În ultima perioadă, odată cu creșterea exponențială a cantității de date disponibile, s-a pus problema aplicării sistemelor de recomandare, cu precădere a celor de tip filtrare colaborativă, în domeniul bio-medical. Problemele de acest gen au mai fost abordate în trecut în învățarea automată, fiind însă considerate probleme clasice de clasificare. Este îndreptățită așadar comparația între filtrarea colaborativă și clasificare. Un studiu efectuat de Hassan și Syed, unde a fost folosit un set de date de 4557 de instanțe, arată că filtrarea colaborativă de tip memory-based dă rezultate superioare algoritmilor Regresie Logistică și SVM [4]. Pe de altă parte, un alt studiu realizat de Hao și Blair, unde s-au folosit cinci seturi de date, arată că filtrarea colaborativă este în mod consistent inferioară algoritmilor de clasificare [2].

Obiectivul acestor aplicații rămâne același: prezicerea de "ratinguri" pentru instanțele neevaluate. Pentru bio-medicină, raționamentul este că pacienții care au caracteristici clinice asemănătoare prezintă un risc asemănător de dezvoltare a unei boli. Translatarea de la problema clasică de filtrare colaborativă se face sub forma următoarelor analogii:

1. utilizatori - pacienti
2. obiecte - caracteristici clinice. Din acest punct de vedere, clasa care trebuie prezisă este tratată ca o caracteristică.

3. ratinguri - valorile caracteristicilor clinice

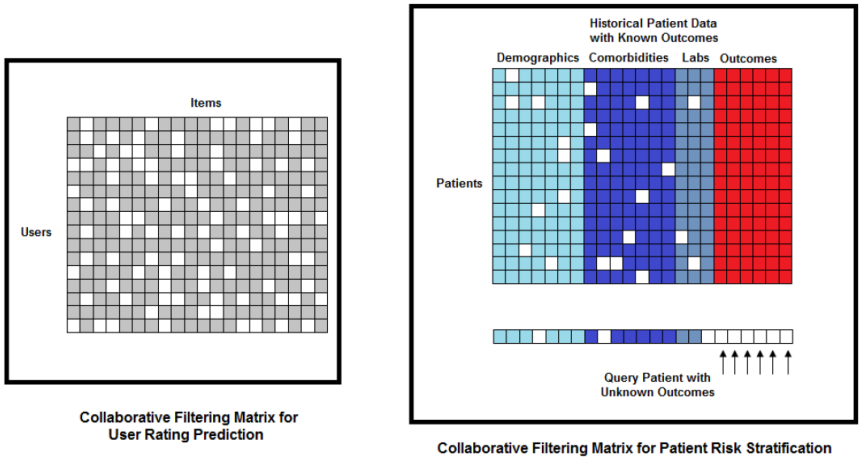

Figura 3.1: *Ilustrare a conversiei filtrare colaborativă - clasificare.*

În continuare vom descrie modalitatea prin care vom efectua experimentul.

3.2 Metodă

3.2.1 Date

Vom aplica cei patru algoritmi pe cinci seturi de date din domeniul medical, pe care le vom descrie în continuare:

1. **Breast Cancer Wisconsin** - conține 699 de observații. Are nouă atribute, fiecare având valori întregi, în intervalul [1, 10]. Atributul *Target* are valoarea 0 dacă tumoarea este benignă, altfel 1.

2. **Chronic Kidney Disease** - conține 400 de observații. Acest set de date are un număr mare de atribute, și anume 25, dintre care 11 sunt atribute continue. Atributul *Target* are valoarea 0 dacă pacientul este sănătos, altfel 1.

3. **Dermatology** - conține 366 de observații. Acest set de date are un număr mare de atribute, și anume 34. Singurul atribut continuu este *Age*, restul luând valori între [0, 3]. Atributul *Target* ia valori între 1-6, fiecare valoare corespunzând uneia dintre următoarele afecțiuni (în ordine) : psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, pityriasis rubra pilaris.

4. **Egyptian Hepatitis** - conține 1385 de observații. Setul de date este compus din 29 de atribute, din care doar 10 sunt discrete. Atributul *Target* exprimă stadiul de hepatită C în care se află un pacient și este cuprins între [1,4].

5. **Heart Disease** - conține 303 observații. Setul de date are 14 atribute, dintre care x sunt continue. Atributul *Target* ia valoarea 0 dacă pacientul este sănătos, altfel 1.

Toate aceste seturi de date sunt disponibile în baza de date *UCI Machine Learning Repository*.

3.2.2 Preprocesare

Discretizare

Sistemele de recomandare sunt proiectate să lucreze cu ratinguri, acestea fiind, prin natura lor, valori categorice sau ordinare. Vom simula această caracteristică prin discretizarea atributelor continue. Pentru fiecare set de date, valoarea maximă pe care o iau atributele categorice va fi folosită pentru a le discretiza pe cele continue. Astfel, dacă setul de date $\mathcal{X}$ are două variabile categorice care iau valorile {1, 2}, respectiv {1, 2, 3}, atunci valorile continue vor fi discretizate în 3 niveluri și vor lua valorile {1, 2, 3}. Pragurile pentru acest procedeu sunt stabilite în funcție de cuantile, pentru a se asigura un echilibru între nivelurile de discretizare.

Așadar, pentru seturile de date Dermatology, Cleveland și HCV am aplicat discretizarea pe 4 niveluri, iar pentru Chronic Kidney Disease am aplicat discretizarea pe 5 niveluri. Setul de date Breast Cancer Wisconsin nu a avut nevoie de discretizare, deoarece toate atributele erau deja categorice.

Imputarea valorilor lipsă

Algoritmii de clasificare nu suportă prezența unor valori nule, astfel încât vom imputa valorile nule ale unui atribut cu valoarea medie a acelui atribut. În cazul algoritmilor de filtrare colaborativă, instanțele care au valori lipsă nu vor fi incluse în setul de antrenament.

Standardizarea atributelor

Avantajul scalării atributelor este de a evita ca atributele definite pe un interval numeric mai mare să le domine pe cele definite pe un interval mic. Deoarece valorile

kernel ale SVM depind în general de produsul unor vectori de trăsături, există pericolul ca atributele cu valori mari să cauzeze probleme numerice [3]. De aceea, vom scala atributele pe intervalul 0, 1. De asemenea, și pentru Regresia Logistică este recomandată scalarea, deoarece regularizarea face ca valoarea care trebuie prezisă să fie dependentă de mărimea atributelor [1]. Nu vom standardiza atributele pentru filtrarea colaborativă.

3.2.3 Implementarea

Implementarea acestui experiment s-a realizat în Python. Pentru algoritmii de clasificare am folosit biblioteca *scikit-learn*, iar pentru cei de filtrare colaborativă am folosit biblioteca *Surprise*, pe care am menționat-o și în capitolul anterior. Pe lângă acestea, am mai utilizat *pandas*, pentru lucrul cu mulțimile de date.

3.2.4 Determinarea acurateței

Vom aplica regresia logistică, SVM, k-NN și SVD pe cele cinci seturi de date. Așa cum am procedat și în capitolul anterior, vom testa acuratețea algoritmilor prin cross-validare pe patru niveluri: [3, 5, 7, 10]. Vom rula algoritmii de 20 de ori, păstrând apoi media pentru fiecare nivel.

Vom folosi din nou RMSE ca metrică de evaluare.

3.2.5 Reglarea hiper-parametrilor

Vom încerca optimizarea tuturor algoritmilor folosiți prin reglarea hiper-parametrilor. Ca în exemplul aplicativ, ne vom folosi de GridSearch pentru a căuta valorile optimale pentru parametrii. Pentru fiecare algoritm, vom specifica valorile rezultate.

3.3 Rezultate

Pentru început, vom aplica algoritmii neoptimizați. Pentru fiecare set de date, vom compara rezultatele acestora printr-un tabel.

3.3.1 Breast Cancer Wisconsin

nr. niveluri	LR	SVM	k-NN	SVD
3	0.372	0.363	2.215	1.767
5	0.363	0.352	1.968	1.723
7	0.357	0.345	1.900	1.706
10	0.342	**0.333**	1.858	1.697

În urma *GridSearch*-ului am obținut următoarele valori:

- pentru LR: C = 1, penalizarea = L1
- pentru SVM: C = 1, gamma = 0.01, kernel = rbf
- pentru kNN: k = 50, măsura de similaritate = coeficientul Pearson
- pentru SVD: iterații SGD = 5, rata de învățare = 0.002, termenul de regularizare = 0.5

nr. niveluri	LR	SVM	k-NN	SVD
3	0.377	0.354	1.694	1.564
5	0.362	349	1.651	1.496
7	0.352	0.343	1.637	1.450
10	0.338	**0.330**	1.627	1.402

3.3.2 Kidney Disease

nr. niveluri	LR	SVM	k-NN	SVD
3	0.090	0.0539	0.590	0.633
5	0.079	0.0331	0.573	0.605
7	0.074	0.026	0.566	0.596
10	0.065	**0.018**	0.561	0.589

În urma *GridSearch*-ului am obținut următoarele valori:

- pentru LR: C = 0.1, penalizarea = L2
- pentru SVM: C = 1, gamma = 0.01, kernel = rbf
- pentru kNN:k = 50, măsura de similaritate = coeficientul Pearson

- pentru SVD: iterații SGD = 10, rata de învățare = 0.002, termenul de regularizare = 0.4

nr. niveluri	LR	SVM	k-NN	SVD
3	0.078	0.081	0.564	0.630
5	0.067	0.063	0.554	0.602
7	0.065	0.049	0.549	0.597
10	0.059	0.049	0.546	0.557

3.3.3 Heart Attack

nr. niveluri	LR	SVM	k-NN	SVD
3	0.409	0.407	0.690	0.715
5	0.407	0.400	0.679	0.710
7	0.405	0.400	0.675	0.707
10	0.403	**0.394**	0.672	0.705

În urma *GridSearch*-ului am obținut următoarele valori:

- pentru LR: C= 0.01, penalizare = l2
- pentru SVM: C = 0.1, gamma = 0.001, kernel = linear
- pentru kNN: k = 40, măsura de similaritate = coeficientul Pearson
- pentru SVD: iterații SGD = 5, rata de învățare = 0.002, termenul de regularizare = 0.5

nr. niveluri	LR	SVM	k-NN	SVD
3	0.396	0.402	0.684	0.710
5	0.393	0.400	0.675	0.708
7	0.390	0.395	0.672	0.702
10	0.387	0.392	0.671	0.680

3.3.4 Dermatology

nr. niveluri	LR	SVM	k-NN	SVD
3	0.313	0.328	0.601	0.709
5	0.295	0.295	0.574	0.671
7	0.281	0.271	0.566	0.657
10	0.252	**0.238**	0.561	0.648

În urma *GridSearch*-ului am obținut următoarele valori:

- pentru LR: C = 0.1, penalizare = L2
- pentru SVM: C = 10, gamma = 0.001, kernel = linear
- pentru kNN: k = 30, metrica de similitudine = coeficientul Pearson
- pentru SVD: iterații SGD = 10, rata de învățare = 0.002, termenul de regularizare = 0.4

nr. niveluri	LR	SVM	k-NN	SVD
3	0.302	0.337	0.578	0.705
5	0.288	0.288	0.558	0.672
7	0.250	0.240	0.550	0.652
10	0.238	**0.194**	0.546	0.644

3.3.5 Egyptian Hepatitis

nr. niveluri	LR	SVM	k-NN	SVD
3	1.585	1.594	0.961	0.996
5	1.590	1.586	0.4805	0.997
7	1.588	1.589	0.4804	0.997
10	1.591	1.587	0.4803	0.996

- pentru LR: C = 0.1 penalizarea = L2
- pentru SVM: C = 1, gamma = 0.1, kernel = rbf
- pentru kNN: k = 50, metrica de similaritate = coeficientul Pearson
- pentru SVD: iterații SGD = 5, rata de învățare = 0.002, termenul de regularizare = 0.4

nr. niveluri	LR	SVM	k-NN	SVD
3	1.579	1.587	0.874	0.967
5	1.580	1.582	0.479	0.967
7	1.576	1.588	0.479	0.967
10	1.575	1.580	0.476	0.967

Așa cum se poate observa, algoritmul kNN este cel mai performant pe mulțimea de date Egyptian Hepatitis. În rest, SVM oferă cele mai bune rezultate, urmat de Regresia Logistică. Aceste rezultate arată că pe aceste seturi de date, filtrarea colaborativă este inferioară clasificării.

Concluzii

Am început această lucrare prin a prezenta clasele de algoritmi de clasificare, urmați de cei de filtrare colaborativă. Am aplicat acești algoritmi pe exemple clasice, pentru a observa cum se comportă în condiții normale.

Am continuat apoi prin compararea celor două clase de algoritmi pe probleme de clasificare din domeniul medical, deoarece aplicarea filtrării colaborative pe aceste probleme pare să devină o tendință.

Rezultatele la care am ajuns arată că, în continuare, pentru problemele de clasificare, algoritmii clasici par să fie mai eficienți. Cu toate acestea, există și studii care arată contrariul. Trebuie ținut cont de faptul că mulțimile de date pe care am lucrat conțin o cantitate relativ mică de date, precum și că nu sunt cele mai actuale. Astfel, nu trebuie respinsă complet ipoteza că filtrarea colaborativă ar putea fi folosită pentru prezicerea diagnosticelor medicale.

Bibliografie

[1] Jerome Friedman, Trevor Hastie **and** Robert Tibshirani. *The elements of statistical learning*. **volume** 1. 10. Springer series in statistics New York, 2001.

[2] Fang Hao **and** Rachael Hageman Blair. "A comparative study: classification vs. user-based collaborative filtering for clinical prediction". **in**: *BMC medical research methodology* 16.1 (2016), **page** 172.

[3] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin **andothers**. *A practical guide to support vector classification*. 2003.

[4] Shahzaib Hassan; Zeeshan Syed. "From Netflix to Heart Attacks: Collaborative Filtering in Medical Datasets". **in**: (2010).

[5] Regina Barzilay Tommi Jaakkola. *Introduction to Machine Learning-Draft Notes by Topic*. 2015.