Universitatea din Bucureşti

Facultatea de Matematică şi Informatică

Specializarea Inteligenţă artificială

Îmbunătăţirea unui sistem de recomandare

LUCRARE DE DISERTAŢIE

Coordonator ştiinţific

Conf. Dr. Bogdan Alexe

Absolvent

Adrian Ispas

Bucureşti, România

Iunie 2019

Abstract

Abstractul în limba română.

Abstract

Abstractul în limba engleză.

Cuprins

Listă de figuri

Listă de tabele

Capitolul 1

Introducere

1.1 Motivaţie

Volumul de date creşte semnificativ de la an la an astfel până în 2020 se estimează că pentru fiecare persoană de pe planetă vor fi creaţi în fiecare secundă 1.7 MB de date, ceea ce înseamnă peste 13 milioane de GB creaţi în fiecare secundă în lume.În 2018 în fiecare minut se vizionau peste 97 de mii de ore de conţinut pe Netlfix. Peste 4.3 milioane de videoclipuri erau vizionate pe Youtube. Pe Spotify se ascultau 750 de mii de melodii, iar Amazon pregătea peste o mie de pachete [1].

În România, Netflix pune la dispoziţie 575 de filme şi 208 seriale. În Regatul Unit sunt disponibile 2425 de filme şi 542 de seriale, iar în Statele Unite Ale Americii sunt disponibile 2942 de filme şi 629 de seriale [2]. Amazon oferă cumpărătorilor o gamă cu un total de peste 119 milioane de produse, dintre care 44.2 milioane de cărţi, 10.1 milioane de electronice sau 4.5 milioane de produse realizate manual [3].

Cu cât volumul de date pus la dispoziţie de o platforma este mai mare cu atât este mai mare şi necesitatea unui sistem de recomandare care să vină în ajutorul utilizatorului final pentru a explora mai uşor gama de produse oferită de respectiva platformă. De asemnea, acel sistem de recomandare se vrea a fi îmbunătăţit astfel încât să ofere fiecărui utilizator o experienţă cât mai personalizată prin care să recomande, în cazul platformelor de streaming video, conţinut relevant pentru a fi consumat de utilizatorul final, sau în cazul platformelor de ecommerce, produse pe care utilizatorul ar fi dispus să le cumpere.

În majoritatea cazurilor sistemele de recomandare se bazeaza pe metadatele utilizatorilor, precum: regiunea, vârsta, genul, ce alte produse a accesat sau cumpărat şi metadatele

produselor: categoria din care face parte, ratingul acestuia. La acestea se pot adauga şi alte informaţii precum: ce alte produse a apreciat un alt user cu un profil asemanător.

1.2 Obiective propuse

În majoritatea cazurilor primul contact pe care îl avem cu un clip de pe Youtube, cu un film sau serial de pe Netflix sau un produs de pe Amazon este contactul vizual cu imaginea de prezentare a acelui produs.

Astfel, prezenta lucrare de disertaţie are drept obiectiv principal introducerea în sistemul de recomandare de informaţii vizuale extrase din imaginile de prezentare ale produselor. Informaţiile vizuale sunt reprezentate de clusterele create peste imaginile asociate produselor. Fiecare produs are o imagine de prezentare, iar fiecare imagine are un cluster căruia îi aparţine din intervalul $[1, N]$ unde N este corelat cu numărul de categori de produse din baza de date pe care se execută optimizarea. N poate fi ales şi pe baza altor raţionamente.

Scopul final al acestei abordări fiind acela de a observa evoluţia metricilor de evaluare, în cazul nostru acurateaţea şi precizia@k, atunci când informaţia vizuală este introdusă într-un sistem de recomandare, fiind singura informaţie prezentă exceptând matricea de interacţiuni, dar şi cum se comportă un sistem de recomandare când primeşte această informaţie împreună cu alte informaţii, spre exemplu categoria unui articol.

Acurateţea, în acest context, este definită ca fiind probabilitatea ca un exemplu pozitiv ales în mod aleator să fie clasat mai sus în recomandări decât un exemplu negativ ales în mod aleator. Precizia@k este definită de numărul de exemple pozitive aflate în primele k recomandări.

1.3 Structura lucrării

Prezenta lucrare de disertaţie începe prin detalierea fundamentelor teoretice în capitolul II unde sunt prezentate teoriile ce stau la baza realizării acestei lucrări.

De completat ...

Capitolul 2

Fundamente teoretice

2.1 Sisteme de recomandare

2.1.1 Noţiuni generale

Sistemele de recomandare au scopul de oferi sugestii cât mai relevante de articole utilizatorilor unei platforme pe baza unor strategii. Un sistem de recomandare poate folosi una sau mai multe strategii de recomandare după cum vom vedea în continuare. În cazul în care se folosesc cel puţin două strategii, sistemul de recomandare devine un sistem de recomandare hibrid. Prin folosirea mai multor strategii se urmăreşte ca fiecare strategie să vină în completarea celorlalte strategi cu avantajele sale. De cele mai multe ori, în implementarea unui sistem de recomandare, se foloseşte tehnica de filtrare coloborativă împreună cu o altă strategie de recomandare [4].

2.1.2 Strategii de recomandare

Filtrarea coloborativă

Filtrarea coloborativă se bazează pe faptul că utilizatorii care au în prezent preferinţe similare vor avea şi în viitor preferinţe destul de similare. Această abordare foloseşte ratingurile pe care le dau utilizatorii sau oricare altă formă de a da un feedback, îmi place/nu îmi place, pentru a identifica preferinţele comune dintre grupurile de utilizatori. Odată identificate preferinţele se generează recomandări pe baza similarităţilor dintre utilizatori.

Dezavantajul acestei strategii apare în momentul în care în sistem intră un nou utili-

zator. Datorită faptului că utilizatorul este nou, sistemul nu are un istoric al preferinţelor lui, iar în consecinţă nu îl poate asigna unui grup de utilizatori pe baza preferinţelor [4].

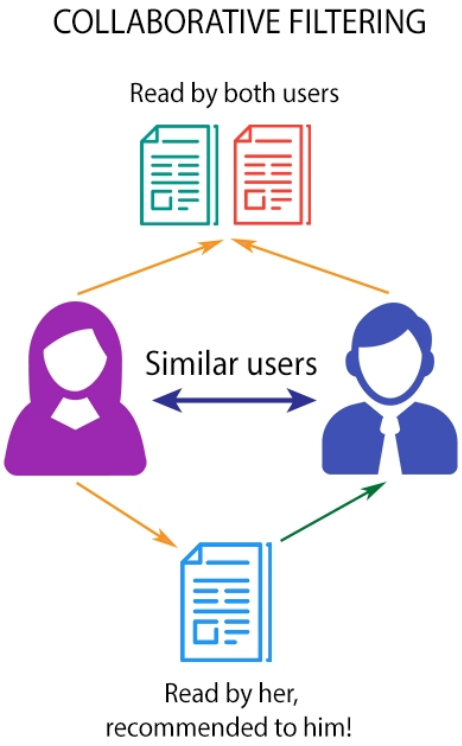


Figura 2.1: Filtrarea coloborativă. Imagine preluată din [5].

Filtrarea bazată pe conţinut

Filtrarea bazată pe conţinut pleacă de la premisa că utiliztorii cărora le-au plăcut articole definite de anumite caracteristici în trecut, vor aprecia aceleaşi tip de articole şi în viitor. Această abordare foloseşte caracteristicile articolelor pentru a le compara cu profilul utilizatorilor şi a oferi recomandări. Calitatea recomandărilor rezultate folosind această strategie este influenţată de setul de caracteristici ales pentru articole. Similar cu filtrarea coloborativă, filtrarea bazată pe conţinut prezintă dezavantaje în momentul în care în sistem intră un nou utilizator fără istoric [4].

Filtrarea demografică

Filtrarea demografică foloseşte atribute precum vârsta, genul, educaţia, etc. pentru a identifica categoriile de utilizatori. Nu prezintă dezavantaje atunci când apar noi utilizatori în sistem şi nu se foloseşte de ratinguri, sau alt sistem de feedback, pentru a face recomandări.

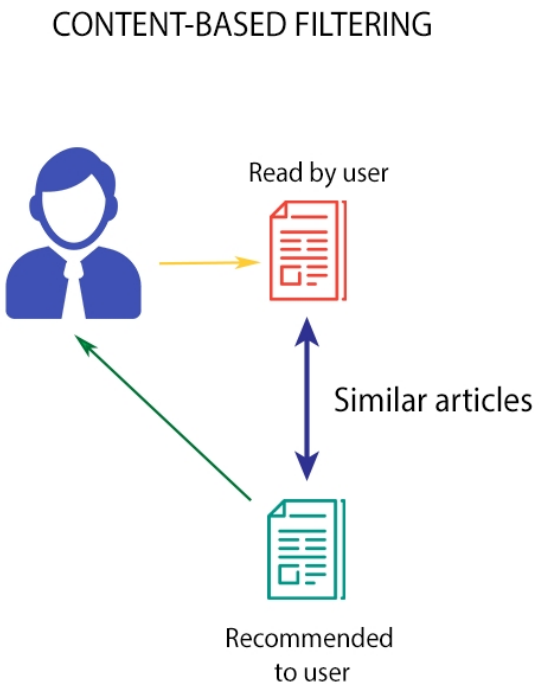


Figura 2.2: Filtrarea bazată pe conţinut. Imagine preluată din [5].

Dezavantajul este reprezentat de faptul că procesul de colectare al datelor demografice poate fi îngreunat de legislaţie, fapt ce reprezintă o limitare a acestei metode [4].

Filtrarea bazată pe cunoştinţe

Filtrarea bazată pe cunoştinţe foloseşte cunoştinţele despre utilizatori şi articole pentru a spune ce articole îndeplinesc cerinţele utilizatorilor şi genereaza recomandări în consecinţă. Filtrare bazată pe cunoştinţe are la bază constrângeri şi este capabilă să recomande chiar şi articole complexe care nu sunt cumpărate atât de des, precum maşini sau case [4].

2.1.3 Funcţii de eroare

BPR: Bayesian Personalised Ranking

Este o metodă ce se bazează pe feedback implicit (click-uri, ratinguri, achiziţii, vizualizări). Exită multe metode ce se bazează pe acest feedback implicit, precum matrix factorization (MF), k-nearest neighbors (kNN), însă acestea nu sunt optimizate pentru ranguri. Metoda de învăţare este bazată pe gradientul descendent şi este recomandată

atunci când se doreşte optimizarea acurateţii.

Definim în continuare U ca fiind mulţimea de utilizatori şi I ca fiind mulţimea de articole. Feedback-ul implicit este reprezentat de mulţimea $S \subseteq U \times I$. De asemenea, definim $I_u^+ := i \in I : (u,i) \in S$ şi $U_i^+ := u \in U : (u,i) \in S$.

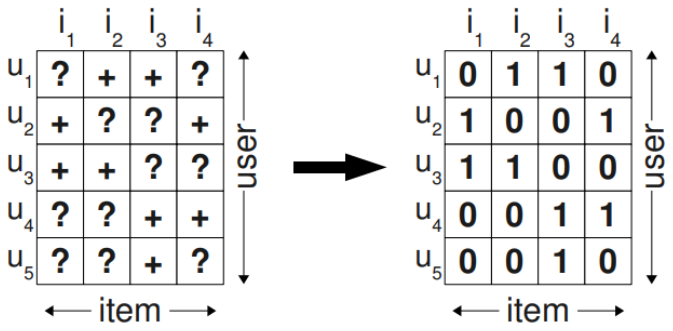


Figura 2.3: Matricea de interacţiuni, mulţimea S. Imagine preluată din [9].

O abordarea uzuală pentru recomandarea de articole este să fie estimat scorul $\hat{x}_{ui}$ care să reflecte preferinţa utilizatorului u pentru articolul i. Apoi fiecare articol primeşte un rang după sortarea scorurilor.

Setul de antrenare (vezi figura 2.4) este definit de mulţimea $D_S := \{(u,i,j)|i \in I_u^+ \wedge j \in I \backslash I_u^+\}$ unde (u,i,j) înseamnă că utilizatorul u preferă articolul i în detrimentul articolului j.

Criteriul de optimizare pentru pentru rangurile personalizate este definit după cum urmează:

$$BPR - OPT := \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \tag{2.1}$$

unde σ este funcţia sigmoid, $\sigma(x) := \frac{1}{1+e^{-x}}$, Θ reprezintă vectorul parametru al modelului care defineşte interacţiunea dintre utilizatorul u, articolul i şi articolul j, iar λ_Θ reprezintă parametrii de regularizare.

Cu aceste definiţi putem defini şi procedura de învăţare a BPR după cum urmează în figura 2.5:

WARP: Weighted Approximate-Rank

Această metodă îşi are originile în procesarea imaginilor şi anume pentru un set de reprezentări ale unor imagini $x \in R^d$ şi pentru un set de reprezentări ale unor adnotări

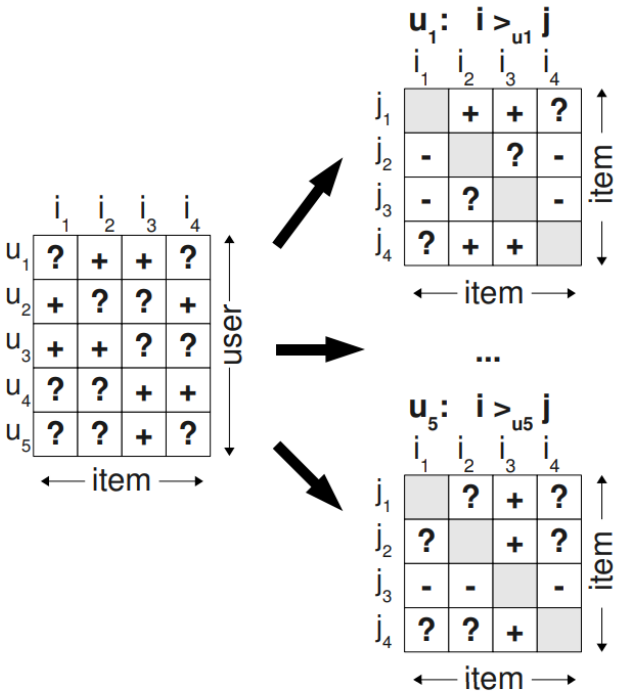


Figura 2.4: Setul de antrenare. + reprezintă articolele i pe care utilizatorul le preferă în locul articolelor j, − utilizatorul preferă articolele j în loc de i, iar ? reprezintă lipsa informaţiei despre acea interacţiune. Imagine preluată din [9].

1: **procedure** LEARNBPR(D_S, Θ)
2: initialize Θ
3: **repeat**
4: draw (u, i, j) from D_S
5: $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
6: **until** convergence
7: **return** $\hat{\Theta}$
8: **end procedure**

Figura 2.5: Optimizarea modelului bazată metoda gradientului descendent cu parametrul de învăţare α şi regularizarea λ_Θ. Imagine preluată din [9].

$i \in \Upsilon = \{1, ..., Y\}$ - inidici intr-un dicţionar cu posibile adnotări, metoda învaţă să mapeze imagini din spaţiul reprezentărilor într-un spaţiu comun R^D

$$\Phi_I(x) : R^d \rightarrow R^D \tag{2.2}$$

în acelaşi timp învăţând şi mapări pentru adnotări în acelaşi spaţiu

$$\Phi_W(i) : 1, ..., Y \rightarrow R^D \tag{2.3}$$

Scopul principal fiind acela de a oferi ranguri posibilelor adnotări pentru o imagine dată astfel încât cel mai mare rang să descrie cel mai bine conţinutul semnatic al imaginii.

Modelul folosit este definit în continuare:

$$f_i(x) = \Phi_W(i)^T \Phi_I(x) \tag{2.4}$$

Metoda învaţă să producă ranguri optimizate pentru primele adnotări din listă, ceea ce înseamnă că optimizează precizia@k.

În ceea ce priveşte funcţia de eroare definim: $f(x) \in R^Y$ ce produce un scor pentru fiecare etichetă şi unde $f_i(x)$ este valoarea etichetei i. Definim funcţia de eroare pentru ranguri ca fiind:

$$err(f(x), y) = L(rank_y(f(x))) \tag{2.5}$$

unde $rank_y(f(x))$ este rangul etichetei corecte data de $f(x)$:

$$rank_y(f(x)) = \sum_{i \neq y} I(f_i(x) \geq f_y(x)) \tag{2.6}$$

unde I este funcţia indicator, iar $L(\cdot)$ transformă rangul în penalizare

$$L(k) = \sum_{j=1}^{k} \alpha_j, \quad cu \quad \alpha_1 \geq \alpha_2 \geq ... \geq 0. \tag{2.7}$$

$L(\cdot)$ poate lua diferite forme în funcţie de ce se doreşte a optimiza: $\alpha_j = \frac{1}{Y-1}$ optimizează rangul mediu, $\alpha_j = 1$ şi $\alpha_{j>1} = 0$ optimizează proporţia de ranguri corecte aflate în top, iar valorile mari ale lui α optimizează primele k în lista de ranguri[8].

Cu definiţiile prezentate mai sus putem descrie algoritmul acestei metode după cum urmează.

Algorithm 1 Online WARP Loss Optimization

Input: labeled data (x_i, y_i), $y_i \in \{1, \dots, Y\}$.
repeat
  Pick a random labeled example (x_i, y_i)
  Let $f_{y_i}(x_i) = \Phi_W(y_i)^\top \Phi_I(x_i)$
  Set $N = 0$.
  repeat
    Pick a random annotation $\bar{y} \in \{1, \dots, Y\} \setminus y_i$.
    Let $f_{\bar{y}}(x_i) = \Phi_W(\bar{y})^\top \Phi_I(x_i)$
    $N = N + 1$.
  until $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$ or $N \geq Y - 1$
  if $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$ **then**
    Make a gradient step to minimize:
      $L(\lfloor \frac{Y-1}{N} \rfloor)|1 - f_y(x_i) + f_{\bar{y}}(x_i)|_+$
    Project weights to enforce constraints (2)-(3).
  end if
until validation error does not improve.

Figura 2.6: Online WARP Loss Optimization. Imagine preluată din [8].

2.2 Reţele neurale convoluţionale

2.2.1 Noţiuni generale

Reţelele neurale convoluţionale sunt reţelele formate din neuroni ce învaţă ponderi (w) şi baiasuri (b). Scopul reţelei convoluţionale este de a primi o imagine la input şi de a scoate la output un scor pentru fiecare clasă ce corespunde imaginii.

Spre exemplu, la input se dă o imagine cu un autovehicul, iar reţeaua convoluţională poate spune că în imagine este o maşină în proporţie de 80%, un camion în proporţie de 10%, un avion în proporţie de 6%, o barcă în proporţie de 3% sau un cal în proporţie de 1%.

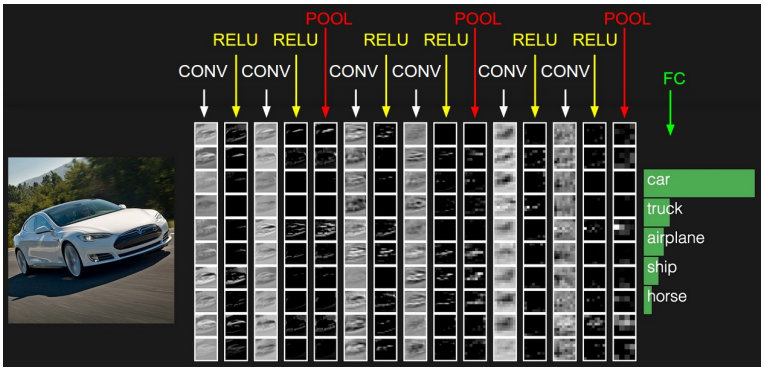


Figura 2.7: Exemplu de reţea convoluţională care primeşte la input o imagine şi produce la output o listă de clase ce pot descrie imaginea de input. Imagine preluată din [7].

Reţelele convoluţionale sunt compuse dintr-o secvenţă de straturi ce poate fi împărţită în trei tipuri principale [7]:

1. Stratul convoluţional este stratul de bază într-o reţea. Parametrii acestui strat sunt reprezentaţi de filtre învăţabile, unde fiecare filtru reprezintă o mică bucată din imaginea de input. De exemplu, un filtru pentru acest strat poate avea dimensiunea de $5 \times 5 \times 3$, dimensiune ce reprezintă faptul că se iau 5 pixeli pe laţime, 5 pixeli pe înălţime şi o adâncime de 3 pixeli, unde adâncimea reprezintă canalele RGB. În continuare se glisează fiecare filtru peste input şi se compune produsul dintre filtre şi input la fiecare poziţie. În urma acestei operaţii se produce un vector de activare 2-dimensional care reprezintă răspunsul filtrului la fiecare poziţie. Altfel spun, reţeaua va învăţa filtre care se activează atunci când sunt prezente anumite tipuri de caracteristici, precum culoarea sau orientarea (vezi figura 2.8).

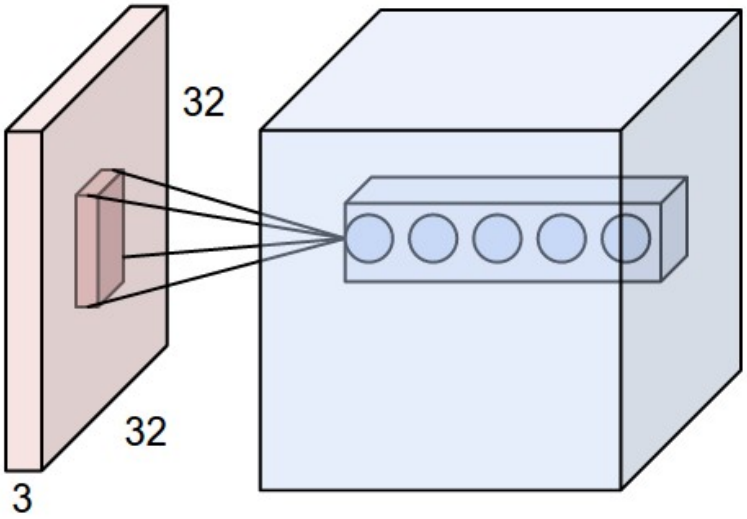
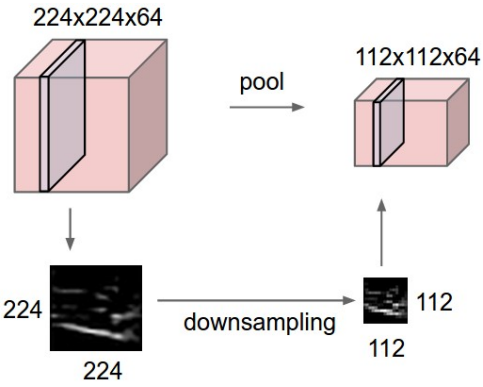
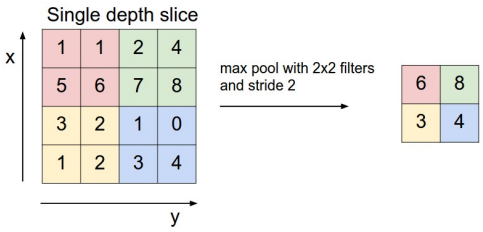


Figura 2.8: Exemplu de filtru aplicat peste input într-un strat convoluţional. Imagine preluată din [7].

2. Stratul de pooling reprezintă o practică des folosită între mai multe straturi convoluţionale succesive. Această operaţie reduce numărul de parametrii (dimensiunea modelului), computaţiile din reţea şi controlează overfittingul. Se execută indepedent pe fiecare nivel al adâncimii unui input şi pastrează valoarea maximă a acelei zone (de cele mai multe ori). Rezultatul este o zonă de caracteristicii mai mică dar care păstrează cea mai relevantă statistică (vezi figura 2.9).



(a) Reducerea dimensiunii.



(b) Filtrul de 2×2 aplicat ce păstrează valoarea maximă.

Figura 2.9: Exemplu de pooling. Imagine preluată din [7].

3. Fully-Connected Layer este stratul în care caracteristicile sunt vectorizate pentru a putea fi folosite.

2.2.2 VGG

VGG este o arhitectură clasică de reţea cu filtre convoluţionale foarte mici, de dimensiune 3×3 şi care poate avea un număr a straturilor de ponderi de 16 - 19.

În ceea ce priveşte arhitectura (vezi figura 2.10), inputul în reţeaua convoluţională este de dimensiune fixă şi anume 224×224 imagine RGB. Mai departe, imaginea este trecută printr-un set de straturi convoluţionale unde sunt utilizate filtre de dimensiune mică, 3×3 - fiind cea mai mică dimensiune ce poate captura noţiunile de stânga/dreapta, sus/jos sau centru. Într-una dintre configuraţii se utilizează un filtru convoluţional de dimensiune 1×1. Pasul în straturile convoluţionale este fixat la 1 pixel.

Poolingul este compus din cinci straturi de max-pooling care urmează după unele straturi convoluţionale. Max-poolingul este calculat cu ferestre de 2×2 pixel şi cu pas de 2 pixeli.

Odată trecută imaginea prin straturile convoluţionale şi cele de pooling ajunge în trei straturi fully-connected. Primele două straturi au câte 4096 de canale fiecare, iar al treilea are 1000 de canale. Canalele celui de-al treilea strat sunt asociate claselor, fiecare canal reprezintă o clasă.

Ultimul strat din reţea este un strat soft-max [10].

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 2.10: Configuraţii ale reţelei VGG. Imagine preluată din [10].

2.2.3 InceptionV3

Prima arhitectura de Inception a apărut sub numele de GoogLeNet. O a doua versiune de Inception a fost definită prin introducerea de batch-uri normalizate. Iar mai apoi, versiunea a treia în care a fost adăugate idea de factorizare.

Factorizarea în filtre convoluţionale mici (vezi figura 2.11) presupune înlocuirea stratului cu filtru de dimensiune 5×5 cu două straturi de dimensiune 3×3 astfel reducânduse dimensiunea de la $5 \times 5 = 25$ la $3 \times 3 + 3 \times 3 = 18$.

Factorizarea spaţială în convoluţii asimetrice (vezi figura 2.12) presupune înlocuirea stratului cu filtru de dimensiune 3×3 cu două straturi de dimensiune 3×1 şi 1×3 astfel reducânduse dimensiunea de la $3 \times 3 = 9$ la $3 \times 1 + 1 \times 3 = 6$.

Clasificatorul auxiliar (vezi figura 2.13) este utilizat în InceptionV3 ca regulizator şi este poziţionat în partea superioară a utimelor 17×17 straturi. Batch-urile normalizate sunt de asemeanea folosite în clasificatorul auxiliar.

Reducerea eficientă a dimensiunii (vezi figura 2.14) se face prin utilizarea a două

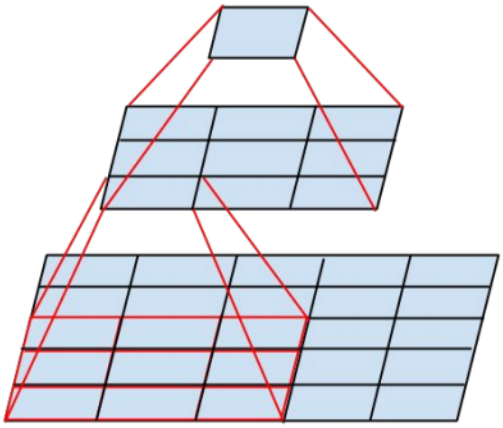


Figura 2.11: Factorizarea în filtre convoluţionale mici. Filtrul de dimensiune 5×5 înlocuit cu două de dimensiune 3×3. Imagine preluată din [14].

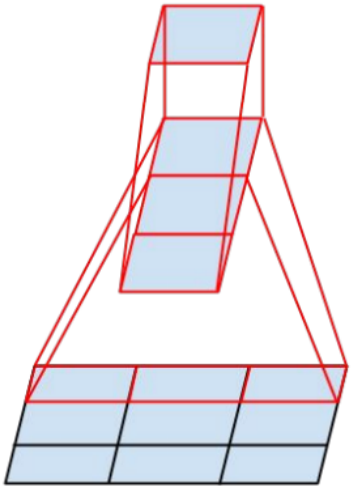


Figura 2.12: Factorizarea în filtre convoluţionale mici. Filtrul de dimensiune 3×3 înlocuit cu două de dimensiune 3×1 şi 1×3. Imagine preluată din [14].

blocuri paralele, fie P şi C. Primul dintre acestea, P, fiind un strat de activare de pooling (media sau maxim pooling). Ambele straturi au filtre cu pas 2 care sunt concatenate.

Arhitectura completă este prezentată în figura 2.15.

2.2.4 ResNet

Fie $H(x)$ maparea de bază unde x reprezintă inputul. Funcţia reziduală poate fi aproximată cu $F(x) := H(x) - x$, maparea de bază fiind $F(x) + x$.

Învăţarea reziduală se aplică la câtva grupuri de straturi. Putem defini un bloc de

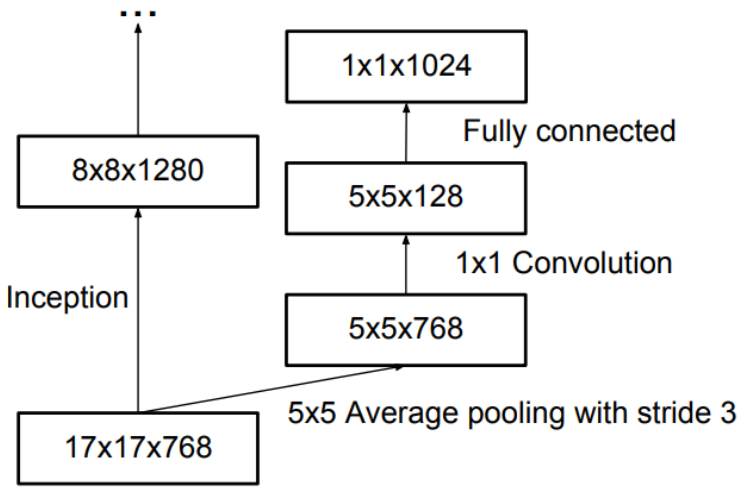


Figura 2.13: Clasificatorul auxiliar. Imagine preluată din [14].

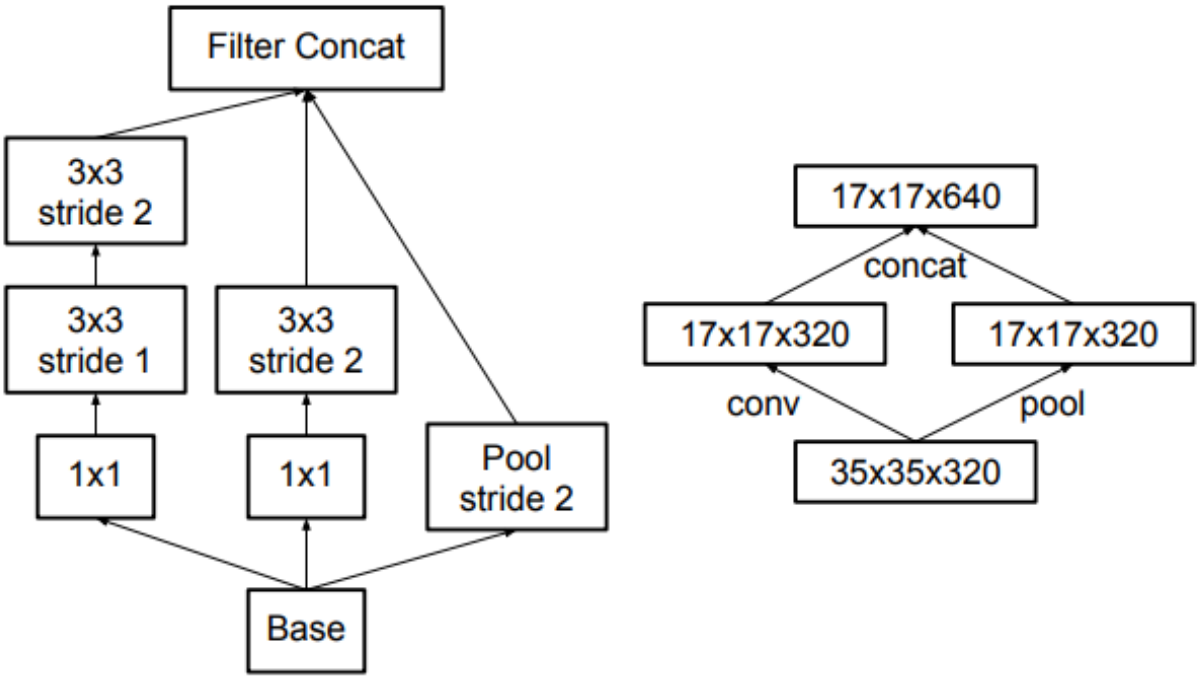


Figura 2.14: Modulul care reduce dimensiunea. Diagrama din dreapta reprezintă aceaşi soluţie însă din perspectiva dimensiunii reţelei. Imagine preluată din [14].

straturi ca fiind

$$y = F(x, \{W_i\}) + x \tag{2.8}$$

unde x şi y reprezintă inputul şi outputul straturilor considerate. Funcţia $F(x, \{W_i\})$ reprezintă maparea reziduală ce trebuie învăţată.

Dimensiunea lui x şi F din ecuaţia de mai sus trebuie să fie egale. Redefinim ecuaţia

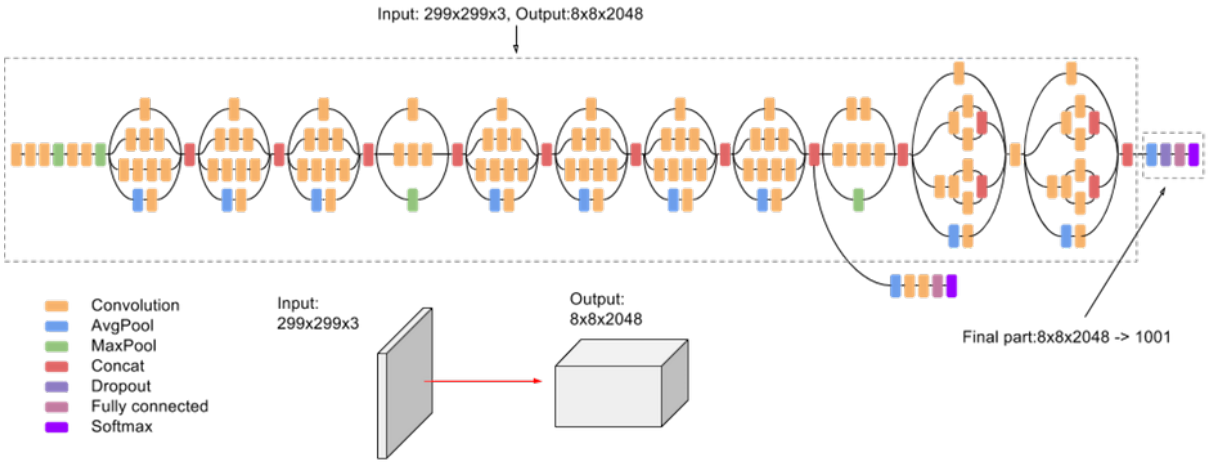


Figura 2.15: Arhitectura reţelei InceptionV3. Imagine preluată din [14].

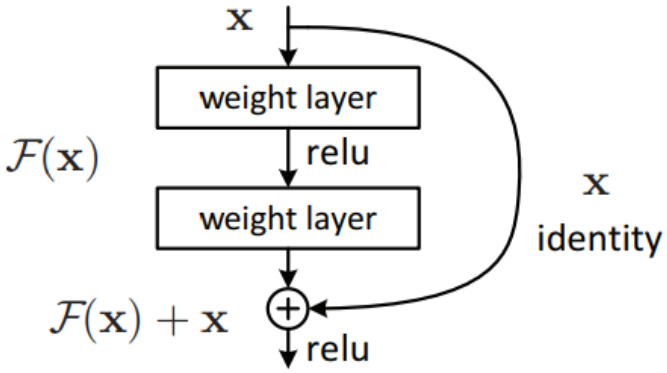


Figura 2.16: Învăţarea reziduală. Imagine preluată din [11].

după cum urmează

$$y = F(x, \{W_i\}) + W_s x \tag{2.9}$$

unde W_s este o proiecţie liniară a scurtăturilor conexiunilor pentru ca dimensiunile să se potrivească.

ResNet (vezi figura 2.17) pleacă de la o reţea simplă. Reţeaua simplă fiind inspirată de reţeaua VGG. Straturile convoluţionale au în general filtre de dimensiune 3×3 şi se bazează de două reguli de design: - pentru outputuri cu acelaşi număr de caracteristici, straturile vor avea acelaşi număr de filtre; - dacă numărul de caracteristicii este injumătăţit, numărul de filtre este dublat astfel încât să fie păstrată complexitatea de timp pe strat.

Poolingul se realizează dupa straturile convoluţionale cu un pas de 2 pixeli. Reţeaua se termină cu un strat de pooling mediu şi un strat fully-connected softmax cu 1000 de canale.

Bazată pe reţeaua descrisă mai sus, reţeaua reziduală presupune inserţia unor scurtături. Scurtăturile identice (vezi formula 2.8) pot fi direct utilizate când inputul şi outputul au aceaşi dimensiune. Când dimensiunea creşte considerăm două opţiuni: - scurtătura calculează în continuare maparea identităţii. Această opţiune nu introduce parametrii noi; - proiecţia scurtăturii din formula 2.9 este utilizată pentru a potrivi dimensiunile. În ambele situaţii când se folosesc scurtăturile pentru pentru a sări peste două straturi sunt calculate cu un pas de 2 pixeli.

2.2.5 NASNet

NASNet este o arhitectură de reţea bazată pe tehnica de căutare Neural Architecture Search (NAS, figura 2.18). NAS presupune un controler cu o reţea neurală recurentă care conţine mai multe reţele copii cu arhitecturi diferite. Reţelele copii sunt antrenate să conveargă pentru a obţine o anumită precizie pe un set de antrenare. Rezultatele sunt utilizate pentru a actualiza controlerul ceea ce înseamnă că acest controler va genera arhitecturi mai bune în timp.

Plusul principal pe care îl aduce reţeaua NASnet este reprezentat de proiectarea unui nou spaţiu de căutare astfel încât cea mai bună arhitectură pe setul de date CIFAR-10 poate scala către rezoluţii ale imaginilor cât mai mari într-un interval definit. Astfel, acest spaţiu poartă numele de *NASNet search space*. În abordarea NASNet, arhitecturile reţelelor convoluţionale sunt manual predeterminate, fiind compuse din celule convoluţionale repetate de multe ori unde, fiecare celulă convoluţională are aceaşi arhitectură dar ponderi diferite.

Pentru a construi mai uşor arhitecturi scalabile pentru imagini de orice dimensiune este nevoie de două tipuri de celule convoluţionale pentru a îndeplini două funcţii principale: - celule convoluţionale care returnează o hartă de caracteristici cu aceaşi dimensiune. Acest tip de celule se numeşte *Celulă Normal*; - celule convoluţionale care returnează o hartă de caracteristici cu înălţimea şi lungimea harţii divizată cu un factor doi. Acest tip de celule se numeşte *Celulă de reducere* (vezi figura 2.19).

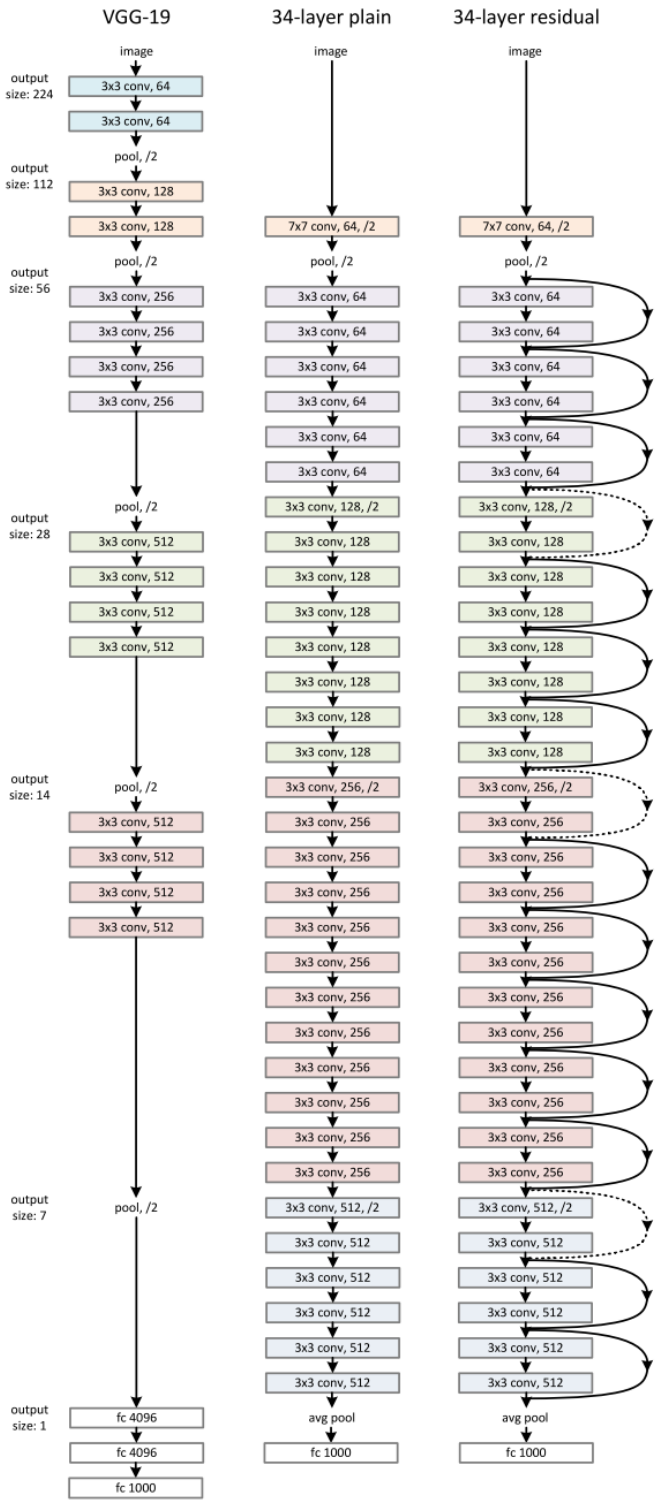


Figura 2.17: Prima reţea (stânga) este o reţea VGG19. A doua reţea (centru) este o reţea simplă cu 34 de straturi. A treia reţea (dreapta) este o reţea reziduală cu 34 de straturi. Imagine preluată din [11].

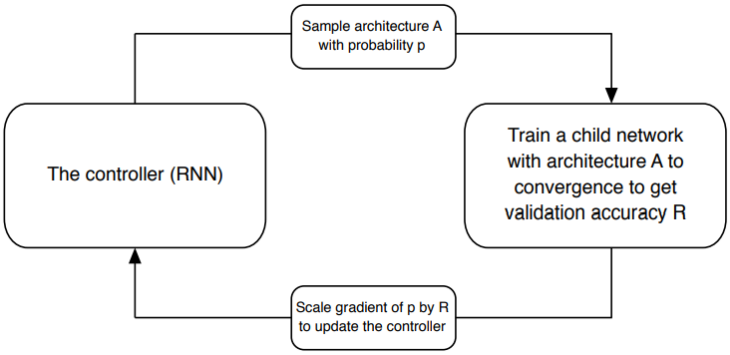


Figura 2.18: Privire de ansamblu asupra unei Neural Architecture Search. Imagine preluată din [13].

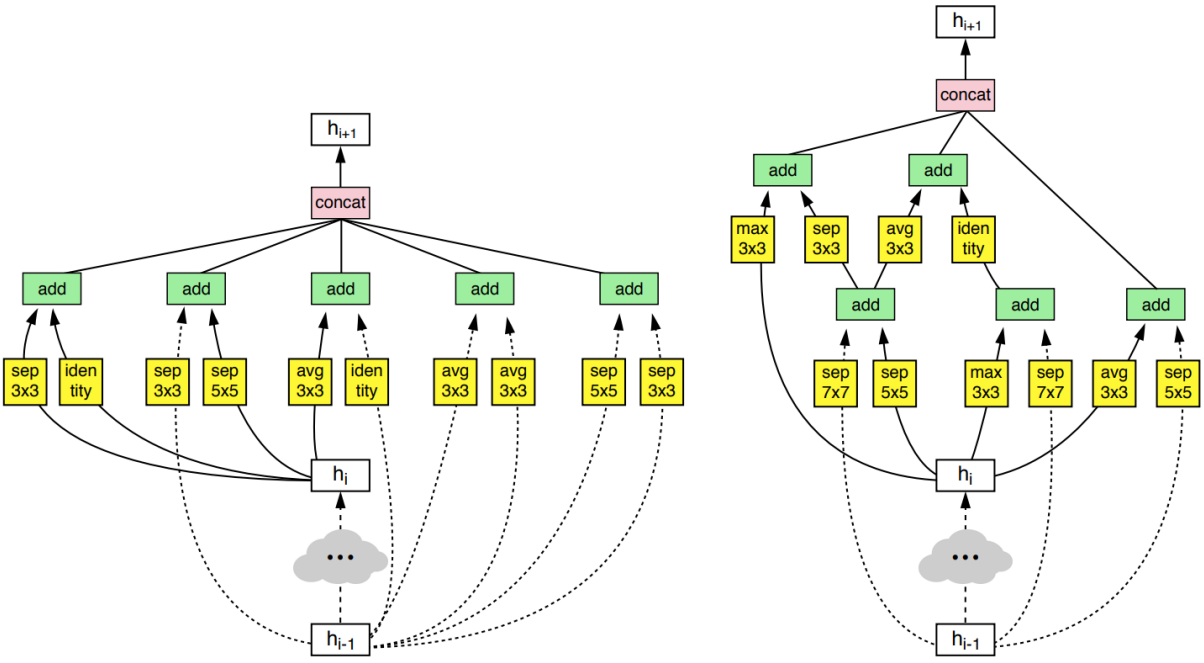


Figura 2.19: Celule normale (dreapta). Celule de reducere (stânga). Imagine preluată din [13].

2.3 Clustere

2.3.1 Noţiuni generale

Clusterizarea este un proces de grupare a unor articole în sensul în care articolele din acelaşi cluster sunt foarte similare între ele din punct de vedere al caracteristicilor. Această metodă este des utilizată în data mining, analiza datelor, machine learning, recunoaşterea tiparelor sau regăsirea informaţiei. Există mai multe tipuri de algoritmi de clusterizare, însă, ideea de bază este aceaşi: clusterele sunt grupuri cu distanţe foarte mici între membrii grupului (fie distanţa euclidiană, de exemplu).

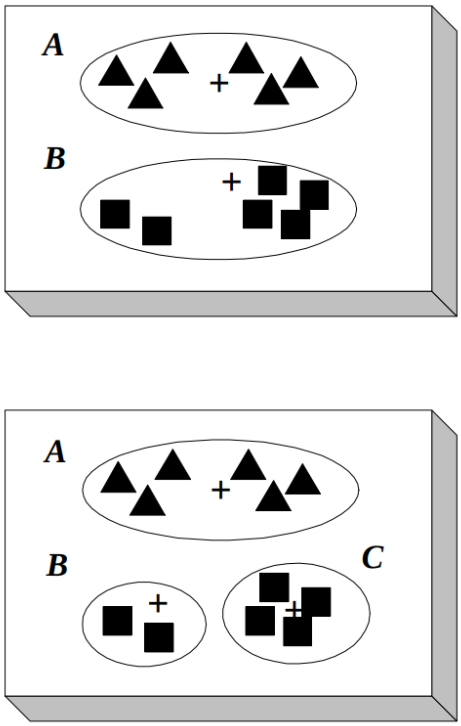


Figura 2.20: Exemplu de două clustere (sus). Exemplu de trei clustere (jos). Imagine preluată din [15].

Procesul de clustering poate fi împărţit în etape după cum urmează [15]:

1. Colectarea datelor: alegerea articolelor pentru care se va aplica clusterizarea;
2. Screening-ul iniţial: presupune extragerea caracteristicilor relevante pentru fiecare articol din dataset;
3. Reprezentarea: presupune pregătirea datelor pentru a putea fi folosite de către algoritmul de clusterizare, tot aici alegânduse şi măsura de similaritate;

4. Tendinţa de grupare: se verifică dacă datele au o tendinţă naturală de grupare; poate fi sărită pentru baze de date mari;

5. Strategia de clusterizare: se alege algoritmul de clusterizare şi parametrii iniţiali;

6. Validarea: se evaluează manual/vizual sau prin alte metode definite rezultatele obţinute în urma clusterizări;

7. Interpretarea: în această se compară rezultatele pe mai multe clustere, combinaţii de clustere şi se trag concluziile.

2.3.2 K-nearest neighbors

KNN reprezintă un model de clasificare simplu şi eficient în multe cazuri. Pentru ca un articol t să fie clasificat sunt căutaţi cei mai propiaţi k vecini formând regiunea lui t. Cei mai apropiaţi veci sunt căutaţi cu o măsură de similaritate, de obicei distanţa euclidiană sau similaritatea cosinus. Votul majoritar din acea regiune este folosit pentru a decide clasificarea lui t. k-ul este valoarea de care depinde destul de mult rata de succes a calsificării, cea mai simplă metodă de a alege un k optim fiind reprezentată de rularea algoritmului pentru mai multe valori ale lui şi observarea evoluţiei rezultelor.

Fie D o colecţie de n clase cunoscute $\{d_1, d_2, ..., d_n\}$. $Sim(d_i)$ - similaritatea celui mai indepărtat punct din regiunea locală, $N(d_i)$ - numărul de puncte din interiorul unei regiuni locale. Algoritmul de construcţie al modelului este definit după cum urmează [16]:

1. selectăm o măsură de similaritate şi creem o matrice de similaritate peste baza de date de antrenare;

2. setăm toate datele cu eticheta neclasificate;

3. pentru fiecare intrare cu eticheta de neclasificat căutăm cea mai mare regiune locală care acoperă cel mai mare număr de vecini cu aceaşi categorie.

4. căutam intrarea d_i cu cea mai mare regiune N_i printre toate regiunile locale şi creem o reprezentare $< Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i) >$ în modelul M pentru a reprezenta toate intrările acoperite de regiunea N_i şi setăm etichete pentru toate aceste intrări;

5. repetăm paşi 3 şi 4 până când toate intrările din baza de date de antrenare au fost clasificate;

6. modelul M este format din toate reprezentările setate în procesul de învăţatare descris mai sus.

Algoritmul de clasificare este definit după cum urmează:

1. pentru ca o nouă intrare d_t să fie clasificată, calculăm similaritatea ei cu toate celelalte reprezentări din model;

2. dacă d_t este acoperit doar de o reprezentare $< Cls(d_j), Sim(d_j), Num(d_j), Rep(d_j) >$ în sensul că distanţa de la d_t la d_j este mai mică decât $Sim(d_j)$, d_t este astfel clasificat ca făcând parte din clasa lui d_j;

3. dacă d_t este acoperit de două sau mai multe clase, clasificăm d_t ca făcând parte din reprezentarea cu cea mai mare valoare a $Num(d_j)$, adică regiunea care acoperă cel mai mare număr de intrări din baza de date de antrenare;

4. dacă nu există nicio reprezentare în modelul M care să acopere d_t, clasificăm d_t cu o clasă nouă.

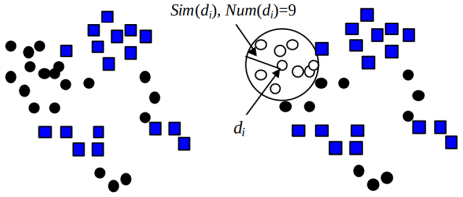
Un exemplu vizual de execuţie a algoritmului de kNN, parcurs pas cu pas, este prezentat în figura 2.21. Exemplu conţine 36 de intrări din 2 clase marcate prin pătrat şi cerc. Datele de test sunt reprezentate prin tringhiuri.

Un al doilea exemplu de clasificare este prezentat în figura 2.22:

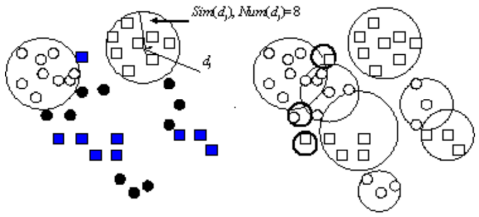
2.3.3 Metrici de evaluare a clusterelor

Coeficientul silhouette

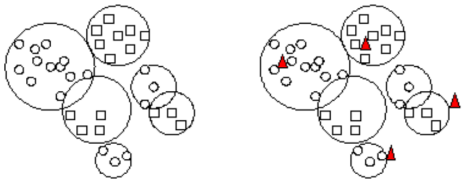
Coeficient (vezi formula 2.10) este folosit pentru a evalua clusterele în învăţarea nesupervizată. Este calculat utilizând distanţa euclidiană medie intra-clustere (a) şi distanţa medie către cel mai apropiat cluster (b) pentru fiecare intrare, adică distanţa dintre o intrare şi cel mai apropiat cluster din care nu face parte. Numărul de etichete trebuie să respecte constrangerea $2 \leq nr_{etichete} \leq nr_{etichete} - 1$. Valorile returnate de acest coeficient sunt cuprinse în intervalul $[-1, 1]$. Valorile apropiate de 0 indică clustere care se suprapun, valorile negative în general indică că există intrări asignate în clusterul greşit, iar



(a) Distribuţia iniţială a datelor (stânga) şi prima reprezentare obţinută (dreapta)

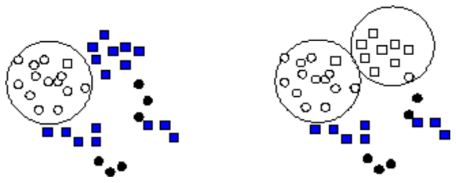


(b) A doua reprezentare obţinută (stânga) şi modelul înainte de triere (dreapta)

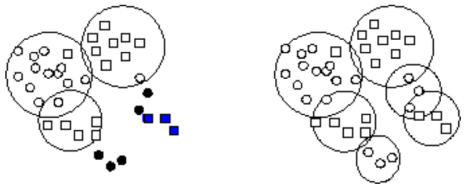


(c) Modelul după triere (stânga) şi distribuţia datelor de test (dreapta)

Figura 2.21: Exemplu 1 de clasificare cu kNN. Imagine preluată din [16].



(a) Prima reprezentare obţinută (stânga) şi a doua reprezentare obţinută(dreapta)



(b) A treia reprezentare obţinută (stânga) şi modelul final (dreapta)

Figura 2.22: Exemplu 2 de clasificare cu kNN. Imagine preluată din [16].

valorile apropiate de 1 indică o separaţie bună intre clustere [17].

$$\frac{b-a}{max(a,b)} \tag{2.10}$$

Bibliografie

[1] Data never sleeps 6.0
`https://www.domo.com/learn/data-never-sleeps-6`

[2] Netflix International: What movies and TV shows can I watch, and where can I watch them?
`https://www.finder.com/global-netflix-library-totals`

[3] How Many Products Does Amazon Sell? – April 2019
`https://www.scrapehero.com/number-of-products-on-amazon-april-2019/`

[4] Erion Çano, Maurizio Morisio. *Hybrid Recommender Systems: A Systematic Literature Review.* Intelligent Data Analysis, vol. 21, no. 6, pp. 1487-1524, 2017

[5] An Overview of Recommendation Systems
`http://datameetsmedia.com/an-overview-of-recommendation-systems/`

[6] LightFM 1.15 - documentation
`http://lyst.github.io/lightfm/docs/lightfm.html`

[7] CS231n: Convolutional Neural Networks for Visual Recognition
`http://cs231n.stanford.edu/2018/syllabus.html`

[8] Jason Weston, Samy Bengio, Nicolas Usunier. *Wsabie: Scaling up to large vocabulary image annotation.* IJCAI. Vol. 11. 2011.

[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. *BPR: Bayesian personalized ranking from implicit feedback.* Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009.

[10] Karen Simonyan, Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* ICLR, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p770-778, 2016.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p2818-2826, 2016.

[13] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. *Learning Transferable Architectures for Scalable Image Recognition.* IEEE/CVF Conference on Computer Vision and Pattern Recognition, p8697-8710, 2018.

[14] Advanced Guide to Inception v3 on Cloud TPU
`https://cloud.google.com/tpu/docs/inception-v3-advanced`

[15] Data Clustering Techniques
`http://www.cs.toronto.edu/~periklis/pubs/depth.pdf`

[16] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, Kieran Greer. *KNN Model-Based Approach in Classification.* Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg, 2003.

[17] Sklearn metrics - silhouette score
`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`