Universitatea Tehnică “Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare

Inteligență artificială
Aplicarea regulii Dempster-Shafer pentru calcularea evidențelor în situații concrete

Studenți:
Buşaga Maria
Radu Daniel
Grupa 1409A

Ianuarie, 2025

Cuprins

Introducere

Ce este Teoria Evidențelor?

Teoria Evidentelor, cunoscută și sub numele de **Teoria Dempster-Shafer** (sau **Teoria Funcțiilor de Credință**), este un cadru matematic folosit pentru a reprezenta și manipula incertitudinea, în special atunci când informațiile disponibile nu sunt complete sau precise.

Este un sistem mai general decât **teoria probabilităților** deoarece permite mai multe grade de incertitudine și mai multe surse de informație care pot fi combinate.

În cadrul teoriei evidentelor, folosim **funcții de credință (belief functions)** pentru a reprezenta credințele, **plauzibilitatea (plausibility)** pentru a reprezenta ceea ce poate fi adevărat pe baza unui set de dovezi și **masele de credință (mass functions)** pentru a distribui credințele în mod distribuit asupra diferitelor subseturi ale unui spațiu de posibilități.

Descrierea problemelor considerate

1. Agregarea polurilor pentru alegeri în încercarea de a prezice rezultatul.

   În cazul exit-pollurilor, rezultatele nu prezic cu exactitate procentul de voturi pe care un partid îl va obține, ci indică mai degrabă ordinea în care vor fi alese partidele, însoțită de un anumit grad de încredere. Din această cauză, aplicarea clasică a regulii Dempster-Shafer nu este direct aplicabilă, deoarece nu este vorba de o combinare simplă a probabilităților, ci de o problemă de combinare a diverselor posibile ordine ale partidelor, fiecare având un anumit grad de încredere asociat. Astfel, problema poate fi abordată din perspectiva combinării maselor de credință pentru fiecare posibilă ordine, ținând cont de incertitudinea și conflictele dintre sursele de informație, reflectând atât ordinea relativă a partidelor, cât și încrederea atribuită fiecărei ordini.

2. Agregarea simptomelor pentru diagnosticarea COVID-19 și a altor boli.

   Problema constă în combinarea informațiilor provenite din simptome (precum febra sau tusea) și alte surse medicale pentru a determina diagnosticul cel mai plauzibil, cum ar fi COVID-19, gripa sau alte afecțiuni. Aceasta implică gestionarea incertitudinii și conflictelor între surse, utilizând regula Dempster-Shafer pentru a atribui un grad de încredere fiecărei ipoteze, reflectând complexitatea diagnosticării în context medical.

Sumar teoretic

Cum funcționează regula Dempster-Shafer?

Regula Dempster-Shafer este folosită pentru a **combina** masele de credință provenite de la diferite surse de informație. Aceasta este importantă atunci când avem mai multe surse care ne furnizează informații incorecte, incomplete sau contradictorii.

Cum combinăm masele de credință?

Regula Dempster-Shafer permite combinarea acestor mase astfel încât să obținem o masă combinată care reflectă toate sursele de informație. Cu toate acestea, unele surse pot adăuga incertitudine și conflict între ele, iar acest conflict trebuie să fie gestionat.

Combinarea maselor de credință:

Formula principală pentru combinarea a două mase de credință m1m_1 și m2m_2 din două surse de informație diferite este:

$$mcombined(X) = \frac{\sum_{A \cap B = X} m1(A) \cdot m2(B)}{1 - Conflict}$$

unde:

- A și B sunt subseturi din spațiul posibilităților (setul de ipoteze care pot fi adevărate).
- m1(A) și m2(B) sunt masele de credință atribuite de cele două surse pentru subseturile A și B.
- **Conflictul** reprezintă suma produselor maselor pentru subseturi care nu se pot intersecta (adică care sunt **incompatibile**).

Explicația pașilor

1. **Identificarea subseturilor compatibile**: Masele de credință din cele două surse sunt combinate doar pentru subseturi care pot coexista. Dacă intersecția a două subseturi este goală (adică sunt incompatibile), acestea nu pot contribui la masă. De exemplu, dacă două surse au credințe diferite, să zicem că una crede că „A” este adevărată și cealaltă că „B” este adevărată, atunci combinarea lor va trebui să reflecte incertitudinea.

2. **Calcularea conflictului**: Conflictul apare atunci când două surse atribuie mase diferite unui subset comun care nu poate fi adevărat simultan. Conflictul este calculat prin suma produselor maselor pentru subseturi incompatibile. Acesta trebuie scăzut din numitorul formulei pentru a obține o combinație corectă.

3. **Numărătorul**: În numărător, se adună produsele maselor de credință pentru toate perechile de subseturi compatibile. Așadar, numărătorul conține toate contribuțiile

posibile ale diferitelor surse care nu sunt contradictorii.

4. **Normalizarea**: Împărțind suma produselor (numărătorul) la *1−Conflict*, obținem masele combinate. Astfel, conflictul reduce masiv valoarea masei combinate, reflectând inconsecvențele dintre surse.

Motivul pentru care funcționează:

1. Flexibilitatea în gestionarea incertitudinii:

Teoria Dempster-Shafer este mult mai flexibilă decât teoria probabilităților pentru că permite lucrul cu **incertitudini parțiale** și nu necesită ca masele de credință să fie complet precise. De exemplu, putem spune că „A sau B este adevărat” fără a trebui să atribui o probabilitate precisă unei anumite ipoteze.

2. Manipularea surselor multiple de informație:

Regula Dempster-Shafer poate combina informații provenite din surse multiple, chiar și atunci când aceste surse sunt parțial contradictorii sau incomplete. Teoria probabilităților clasică nu poate face acest lucru la fel de eficient. În cazul în care două surse sunt complet incompatibile, conflictul va fi ridicat, iar masa combinată va reflecta acest conflict.

3. Conflictul ca indicator al incertitudinii:

Conflictul este o măsură a **inconsistenței** între sursele de informație. Dacă conflictul este mare (adică sursele se contrazic mult), masa combinată va reflecta incertitudinea ridicată. Dacă conflictul este mic sau inexistent (sursele sunt congruente), masa combinată va fi mult mai precisă și va reflecta o credință mai mare în ipoteza combinată.

Exemple de aplicare:

Un exemplu clasic de aplicare a teoriei Dempster-Shafer este în **sisteme expert** sau **decizii pe baza unor surse multiple** de informație (de exemplu, în medicină sau în procese de detectare a erorilor). Să presupunem că avem două surse de informație:

- **Sursa 1** afirmă că "este foarte probabil ca A să fie adevărat".
- **Sursa 2** afirmă că "B este probabil adevărat, dar A este oarecum posibil".

Dacă cele două surse sunt combinate prin regula Dempster-Shafer, se vor lua în considerare atât credințele comune cât și inconsecvențele între ele. Astfel, rezultatul final poate reflecta credințele combinate ale ambelor surse, iar incertitudinea va fi gestionată mai eficient decât într-un sistem strict probabilistic.

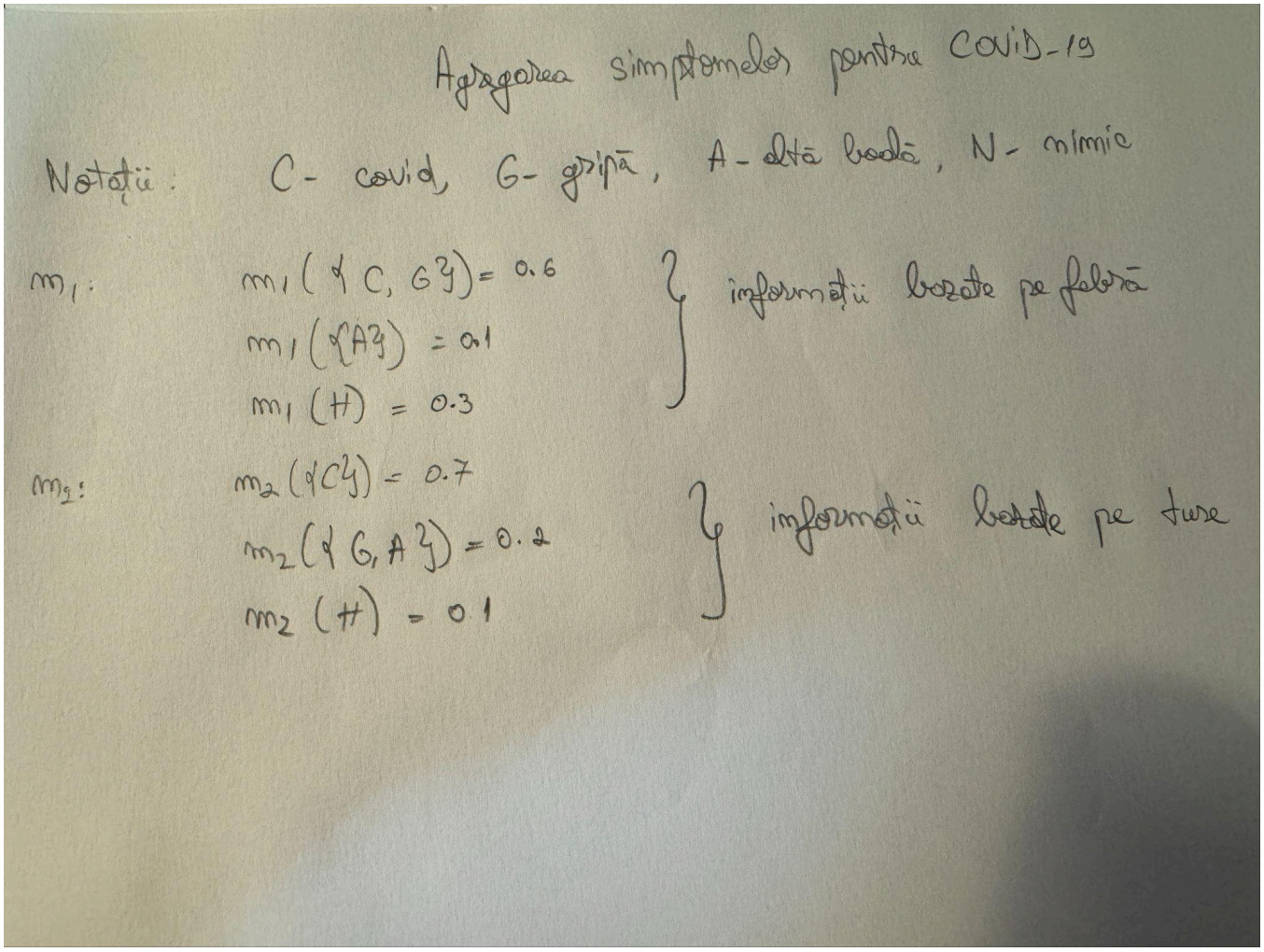
Modalități de rezolvare

Problema 1

În cadrul problemei alegerilor, problema a revenit la a combina evidențele cu mai mult de 3 elemente în evidențe de 2 elemente pentru a păstra ideea de clasament. Spre exemplu, combinația (A, B, C) se transformă în combinațiile (A, B), (A, C), (B, C) toate de masă egală cu cea a grupării inițiale. Masele au fost apoi combinate pentru a putea obține gradul de credință și de plauzibilitate ca fiecare dintre cele 3 partide să iasă câștigător, bazându-ne pe noile evidențe combinate.

După această rearanjare, problema se reduce, mai mult sau mai puțin, la o situație clasică de rezolvare după cum e demonstrat matematic mai jos.

Problema 2



X	$m_1(x)$	Y	$m_2(Y)$	$X \cap Y$	produs
{C, G}	0.6	{C}	0.7	{C}	0.42
{C, G}	0.6	{G, A}	0.2	{G}	0.12
{C, G}	0.6	H	0.1	{G, G}	0.06
{A}	0.1	{C}	0.7	$\emptyset$	0.07
{A}	0.1	{G, A}	0.2	{A}	0.02
{A}	0.1	H	0.1	{A}	0.01
H	0.3	{C}	0.7	{C}	0.21
H	0.3	{G, A}	0.2	{G, A}	0.06
H	0.3	H	0.1	H	0.03

$\Rightarrow K = 0.07$

Numitor = $1 - 0.07 = 0.93$

$m\{C\} = (0.42 + 0.21) / 0.93 = 0.63 / 0.93 = 0.67$

$m\{G\} = 0.12 / 0.93 = 0.129$

$m\{C,G\} = 0.06 / 0.93 = 0.064$

$m\{A\} = (0.02 + 0.01) / 0.93) = 0.0323$

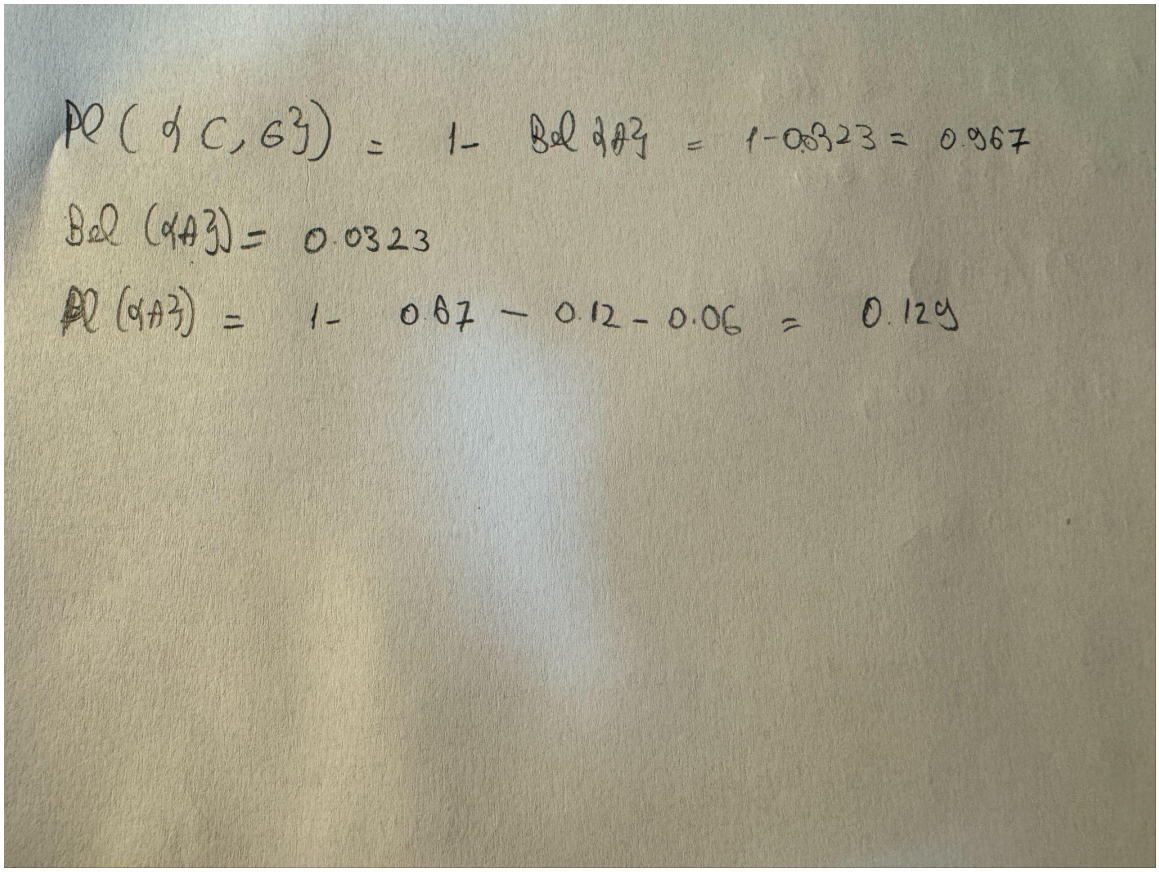
$m\{GA\} = 0.06 / 0.93 = 0.064$

$m\{H\} = 0.03 / 0.93 = 0.032$

$Bel(\{C\}) = 0.67 \Rightarrow Pl(\{C\}) = 1 - 0.129 - 0.032 - 0.064 - 0.032 = 0.749$

$Bel(\{G\}) = 0.129 \Rightarrow Pl(\{G\}) = 1 - 0.67 - 0.323 - [illegible] = 0.2903$

$Bel(\{C,G\}) = 0.67 + 0.129 + 0.064 = 0.87$



Listarea părților semnificative din codul sursă însoțite de explicații și comentarii

Definirea claselor și funcțiilor pentru gestionarea ipotezelor

Python

```
class Hypotheses:
    def __init__(self, parties):
        self.parties = parties

    # Functie pentru generarea combinatiei de ipoteze
    def generate_combinations(self):
        all_combinations = []
        for party in self.parties:
            all_combinations.append((party,))
        # for perm in permutations(self.parties, 2):
```

```
        #     all_combinations.append(perm)
        print("Generated hypotheses (combinations of size 1 and 2):")
        for comb in all_combinations:
            print(f"Combinatia {comb}")
        return all_combinations

    # Functie pentru generarea tuturor submultimilor cu mase
    def generate_all_subsets_with_masses(self, sources):
        all_combinations = {}  # Dictionar pentru combinatii cu mase
        for source in sources:
            for subset, mass in source.items():
                if len(subset) > 1:  # Luam doar combinatiile de
dimensiune 2 sau mai mult
                    combs = combinations(subset, 2)
                    for comb in combs:
                        comb_tuple = tuple(comb)
                        if comb_tuple not in all_combinations:
                            all_combinations[comb_tuple] = 0
                        all_combinations[comb_tuple] += mass  # Adunam
masele
        print("Filtered combinations (only size 2 combinations):")
        for comb, mass in all_combinations.items():
            print(f"Combinatia {comb} are masa {mass:.4f}")  # Afisam
combinatiile filtrate
        return all_combinations
```

În această primă parte a codului, sunt definite clasele și funcțiile necesare pentru manipularea ipotezelor. Clasa `Hypotheses` are două metode principale:

1. `generate_combinations()`: Generează combinații de ipoteze de dimensiune 1, adică fiecare parte a setului de părți este considerată o ipoteză unică. Acesta este un pas preliminar pentru crearea ipotezelor.
2. `generate_all_subsets_with_masses()`: Aceasta este o funcție mai complexă care generează submulțimi de dimensiune 2 și atribuie mase fiecărei combinații bazate pe sursele date, adică sursele sunt dicționare cu mase asociate unor ipoteze. Funcția filtrează doar combinațiile de dimensiune 2, ignorând cele de dimensiune 1.

Definirea calculului maselor combinate și a funcțiilor pentru conflict

Python

```python
# Clasa pentru calculul combinat al maselor in Dempster-Shafer
class ComputeDempsterShafer:
    def __init__(self):
        self.combined_masses = {}  # Mase combinate
        self.conflict = 0  # Conflictele dintre ipoteze

    # Functie pentru calculul intersectiei dintre doua ipoteze
    def compute_intersection(self, h1, h2):
        intersection = tuple(sorted(set(h1) & set(h2)))  # Intersectia a
doua ipoteze
        return intersection if len(intersection) > 0 else None  # Daca
intersectia nu este goala, o returnam

    # Functie pentru actualizarea maselor combinate
    def update_combined_masses(self, m1, m2, hypotheses):
        self.combined_masses = {h: 0 for h in hypotheses}  # Initializam
masele combinate
        self.conflict = 0  # Resetam conflictul
        for h1, m1_val in m1.items():
            for h2, m2_val in m2.items():
                intersection = self.compute_intersection(h1, h2)
                if intersection:  # Daca exista intersectie
                    if intersection not in self.combined_masses:
                        self.combined_masses[intersection] = 0
                    self.combined_masses[intersection] += m1_val * m2_val
# Adunam masele pentru intersectie
                else:
                    self.conflict += m1_val * m2_val  # Adunam conflictul
daca nu exista intersectie

    # Functie pentru normalizarea maselor combinate
    def normalize_combined_masses(self):
        normalization_factor = 1 - self.conflict  # Factor de normalizare
        if normalization_factor <= 0:
            print("Conflict este prea mare; nu este posibila o combinatie
semnificativa.")  # Verificam daca conflictul e prea mare
            return
        for h in self.combined_masses:
            self.combined_masses[h] /= normalization_factor  # Normalizam
masele
```

```
    # Functie pentru calculul maselor combinate
    def calculate_combined_masses(self, m1, m2, hypotheses):
        self.update_combined_masses(m1, m2, hypotheses)  # Actualizam
masele combinate
        self.normalize_combined_masses()  # Normalizam masele
        return self.combined_masses

    # Functie pentru calculul credintei si plauzibilitatii
    def calculate_belief_and_plausibility(self, combined_masses,
hypotheses):
        belief = {h: 0 for h in hypotheses}  # Credinta pentru fiecare
ipoteza
        plausibility = {h: 0 for h in hypotheses}  # Plauzibilitatea
pentru fiecare ipoteza
        for h in hypotheses:
            belief[h] = 0
            for sub_h, mass in combined_masses.items():
                if set(sub_h).issubset(set(h)):  # Verificam daca
sub-ipoteza este inclusa in ipoteza
                    belief[h] += mass  # Adunam masa pentru credinta
            plausibility[h] = 0
            for sub_h, mass in combined_masses.items():
                if set(sub_h) & set(h):  # Verificam daca sub-ipoteza se
intersecteaza cu ipoteza
                    plausibility[h] += mass  # Adunam masa pentru
plauzibilitate
        return belief, plausibility
```

În această secțiune, este definită clasa `ComputeDempsterShafer`, care se ocupă cu calcularea maselor combinate pe baza ipotezelor și surselor date. În această parte a codului, se realizează pașii esențiali pentru combinarea surselor de informații folosind regula de combinare a maselor din Dempster-Shafer:

1. `compute_intersection()`: Calculează intersecția dintre două ipoteze, pentru a determina dacă există o zonă comună între ele.
2. `update_combined_masses()`: Actualizează masele combinate pe baza intersecției dintre ipoteze, adunând masele corespunzătoare. Dacă ipotezele nu se intersectează, se calculează un conflict.

3. `normalize_combined_masses()`: Normalizează masele combinate, având în vedere conflictul calculat, pentru a obține mase valide.
4. `calculate_belief_and_plausibility()`: Calculează credința și plauzibilitatea fiecărei ipoteze pe baza maselor combinate. Aceste două concepte sunt esențiale pentru a înțelege cât de mult susține sursa o ipoteză.

Implementarea motorului de calcul Dempster-Shafer și rularea acestuia

Python

```
# Clasa principala pentru executarea algoritmului Dempster-Shafer
class DempsterShaferEngine:
    def __init__(self, parties, source1, source2, hypotheses=None):
        self.parties = parties  # Partile implicate
        if hypotheses is not None:
            self.hypotheses = hypotheses  # Ipotezele sunt date ca
parametru
            self.source1 = source1  # Sursele pentru primul set de
ipoteze
            self.source2 = source2  # Sursele pentru al doilea set de
ipoteze
        else:
            self.hypotheses = Hypotheses(parties).generate_combinations()
# Generam combinatiile de ipoteze
            self.source1 =
Hypotheses(parties).generate_all_subsets_with_masses([source1])  #
Generam submultimi cu mase pentru sursa 1
            self.source2 =
Hypotheses(parties).generate_all_subsets_with_masses([source2])  #
Generam submultimi cu mase pentru sursa 2
        self.compute_dempster_shafer = ComputeDempsterShafer()  #
Instantiem obiectul care calculeaza Dempster-Shafer

    # Functie pentru rularea procesului
    def run(self):
        combined_masses =
self.compute_dempster_shafer.calculate_combined_masses(self.source1,
self.source2,

self.hypotheses if isinstance(self.hypotheses, list) else
self.hypotheses.keys())
```

```
        belief, plausibility =
self.compute_dempster_shafer.calculate_belief_and_plausibility(combined_
masses,

self.hypotheses if isinstance(self.hypotheses, list) else
self.hypotheses.keys())
        self.print_results(combined_masses, belief, plausibility)

    # Functie pentru afisarea rezultatelor
    def print_results(self, combined_masses, belief, plausibility):
        print("\nCombined and normalized masses:")
        for h, mass in combined_masses.items():
            print(f"{h}: {mass:.4f}")
        print("\nBelief and Plausibility:")
        for h in belief:
            print(f"{h}: Belief = {belief[h]:.4f}, Plausibility =
{plausibility[h]:.4f}")

# Codul principal
if __name__ == "__main__":
    print("Problema 1-Alegeri\n")
    parties = ["A", "B", "C"]
    source1 = {
        ("A", "B", "C"): 0.5,
        ("B",): 0.3,
        ("A", "C"): 0.3
    }
    source2 = {
        ("C", "A", "B"): 0.2,
        ("B", "A", "C"): 0.1,
        ("C", "A"): 0.1
    }
    engine = DempsterShaferEngine(parties, source1, source2)
    engine.run()

    print("\nProblema 2-Covid vs gripa\n")
    H = ('C', 'G', 'A', 'N')
    m1 = {
        ('C', 'G'): 0.6,
        ('A',): 0.1,
```

```
    ('C', 'G', 'A', 'N'): 0.3
}
m2 = {
    ('C',): 0.7,
    ('G', 'A'): 0.2,
    ('C', 'G', 'A', 'N'): 0.1
}
hypotheses = [
    ('C',),
    ('G',),
    ('A',),
    ('C', 'G'),
    ('C', 'G', 'A', 'N')
]
engine = DempsterShaferEngine(H, m1, m2, hypotheses)
engine.run()
```

Rezultatele obținute prin rularea programului în diverse situații, capturi ecran și comentarii asupra rezultatelor obținute

```
Unset
Problema 1-Alegeri

Generated hypotheses (combinations of size 1 and 2):
Combinatia ('A',)
Combinatia ('B',)
Combinatia ('C',)
Filtered combinations (only size 2 combinations):
Combinatia ('A', 'B') are masa 0.5000
Combinatia ('A', 'C') are masa 0.8000
Combinatia ('B', 'C') are masa 0.5000
Filtered combinations (only size 2 combinations):
Combinatia ('C', 'A') are masa 0.3000
Combinatia ('C', 'B') are masa 0.2000
Combinatia ('A', 'B') are masa 0.2000
Combinatia ('B', 'A') are masa 0.1000
Combinatia ('B', 'C') are masa 0.1000
```

```
Combinatia ('A', 'C') are masa 0.1000

Combined and normalized masses:
('A',): 0.4400
('B',): 0.3000
('C',): 0.4400
('A', 'B'): 0.1500
('A', 'C'): 0.3200
('B', 'C'): 0.1500

Belief and Plausibility:
('A',): Belief = 0.4400, Plausibility = 0.9100
('B',): Belief = 0.3000, Plausibility = 0.6000
('C',): Belief = 0.4400, Plausibility = 0.9100
```

('A',):

- **Credința** de 44% indică faptul că, pe baza surselor, există o probabilitate semnificativă ca ipoteza „A” să fie adevărată.
- **Plauzibilitatea** de 91% sugerează că este foarte posibil ca ipoteza „A” să fie adevărată, având în vedere combinațiile cu alte ipoteze.

('B',):

- **Credința** de 30% arată că există o probabilitate mai mică ca „B” să fie adevărată.
- **Plauzibilitatea** de 60% sugerează că există o șansă moderată ca „B” să fie adevărată, însă nu la fel de puternică ca în cazul ipotezei „A” sau „C”.

('C',):

- **Credința** de 44% sugerează o probabilitate relativ mare ca „C” să fie adevărată, similar cu „A”.
- **Plauzibilitatea** de 91% indică faptul că este foarte posibil ca „C” să fie adevărată.

```
Unset

Problema 2-Covid vs gripa


Combined and normalized masses:
('C',): 0.6774
('G',): 0.1290
```

```
('A',): 0.0323
('C', 'G'): 0.0645
('C', 'G', 'A', 'N'): 0.0000
('A', 'G'): 0.0645
('A', 'C', 'G', 'N'): 0.0323

Belief and Plausibility:
('C',): Belief = 0.6774, Plausibility = 0.7742
('G',): Belief = 0.1290, Plausibility = 0.2903
('A',): Belief = 0.0323, Plausibility = 0.1290
('C', 'G'): Belief = 0.8710, Plausibility = 0.9677
('C', 'G', 'A', 'N'): Belief = 1.0000, Plausibility = 1.0000
```

('C',):

- **Credința** de 67,74% sugerează că există o probabilitate mare ca „C” (Covid) să fie adevărată, conform surselor.
- **Plauzibilitatea** de 77,42% arată că este foarte posibil ca „C” să fie adevărată, având în vedere că aceasta este mult mai probabilă decât celelalte ipoteze.

('G',):

- **Credința** de 12,9% indică o probabilitate mică ca „G” să fie adevărată.
- **Plauzibilitatea** de 29,03% sugerează o șansă mai mică ca „G” să fie adevărată.

('A',):

- **Credința** de 3,23% indică o probabilitate foarte mică pentru „A”.
- **Plauzibilitatea** de 12,9% sugerează că nu este foarte posibil ca „A” să fie adevărată.

('C', 'G'):

- **Credința** de 87,1% sugerează că este foarte probabil ca atât „C” cât și „G” să fie adevărate, în combinație.
- **Plauzibilitatea** de 96,77% arată că această combinație este foarte posibilă.

('C', 'G', 'A', 'N'):

- **Credința** și **Plauzibilitatea** de 100% sugerează că această combinație este considerată complet adevărată, ceea ce poate fi o indicație că toate ipotezele au o mare încredere atunci când sunt combinate.

Concluzii

În concluzie, proiectul subliniază utilitatea algoritmului Dempster-Shafer în gestionarea incertitudinii și combinarea informațiilor provenite din surse multiple, permițând evaluarea corectă a ipotezelor în situații complexe. În **Problema 1 - Alegeri**, sursele de informații au generat mase pentru combinațiile de partide politice (A, B, C) și grupările de două partide, iar algoritmul a calculat masele combinate și a estimat credința și plauzibilitatea pentru fiecare partid, evidențiind, de exemplu, că partidul A și partidul C au o probabilitate mai mare de a câștiga, având valori de credință și plauzibilitate de 0.4400 și 0.9100, respectiv. În **Problema 2 - Covid vs Gripă**, sursele au oferit mase pentru combinațiile de ipoteze legate de posibilitatea ca o persoană să fie infectată cu Covid sau gripă, iar algoritmul a calculat masele combinate și a estimat credința și plauzibilitatea acestora, evidențiind că probabilitatea ca o persoană să fie infectată cu Covid este de 0.6774, iar cu gripă de 0.1290. Aceste rezultate demonstrează cum Dempster-Shafer poate fi aplicat eficient pentru combinarea și interpretarea informațiilor incorect definite, contribuind la luarea unor decizii mai bine fundamentate într-o varietate de domenii, de la alegeri politice la diagnostic medical. Algoritmul poate fi extins și adaptat pentru a aborda probleme și mai complexe, având un impact semnificativ în managementul incertitudinii și în luarea deciziilor.

Bibliografie

https://onlinelibrary.wiley.com/doi/10.1155/2014/904596
http://florinleon.byethost24.com/Curs_IA/IA09_Incompletitudine.pdf

Listă cu ce a lucrat fiecare membru al echipei

Radu Daniel- A implementat partea de cod care dezvoltă formulele matematice, pornind de la un exemplu asemănător cu cel din laborator.
Busaga Maria- A extrapolat datele problemei, încercând să obțină rezultate relevante pentru situația problemei alegerilor.