UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAŞI
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DOMENIUL: Calculatoare şi Tehnologia Informaţiei
SPECIALIZAREA: Tehnologia Informaţiei

Inteligență artificială

Rezolvarea problemei de optimizare a unei mașini cu vectori suport cu ajutorul unui algoritm evolutiv

Studenți:
Hușman Carla-Gabriela, 1408A
Pașa Anamaria-Larisa, 1410A
Pîslariu Andreea, 1408A

2023-2023

Cuprins

1. Descrierea problemei considerate

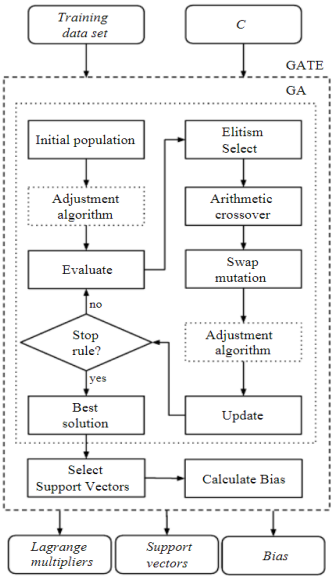
În acest proiect, abordăm o provocare semnificativă în domeniul sănătății: optimizarea diagnosticului de cancer mamar folosind tehnici avansate de învățare automată. Ne concentrăm pe îmbunătățirea performanței SVM (Support Vector Machine) prin integrarea unui algoritm evolutiv cu codare reală, cu scopul de a optimiza atât multiplicatorii Lagrange cât și bias-ul modelului. Acest demers vizează o clasificare mai precisă a tumorilor mamare, contribuind astfel la diagnosticarea și tratamentul mai eficient al cancerului mamar.

Setul de date "Breast Cancer Wisconsin (Original)" [1] este ales ca fundament pentru antrenarea și testarea modelului. Acesta include 699 de instanțe, dintre care 458 (65.6%) sunt maligne și 241 (34.5%) sunt benigne, acoperind un spectru larg de cazuri reale. Fiecare instanță conține 11 atribute, inclusiv grosimea clusturilor, uniformitatea dimensiunii celulelor și a formei, aderența marginală și mărimea nucleului. Aceste caracteristici sunt esențiale pentru identificarea corectă a tipurilor de tumori.

Implementarea se axează pe o metodă de clasificare una versus una, implicând antrenarea unui model SVM unic, care evaluează fiecare pereche de clase în parte. Această abordare simplifică procesul de clasificare, oferind în același timp o acuratețe ridicată.

2. Aspecte teoretice privind algoritmul

Algoritmul evolutiv utilizat în acest proiect îmbină principii ale evoluției naturale cu strategii de optimizare computațională pentru a îmbunătăți clasificarea datelor în cazul cancerului mamar. Inspirat de selecția naturală, algoritmul propune o populație de soluții (cromozomi) care evoluează prin aplicarea repetată a operatorilor genetici: selecție, încrucișare și mutație. Aceta poate juca un rol semnificativ în îmbunătățirea performanței modelelor complexe, cum ar fi Support Vector Machine (SVM), o tehnică puternică de învățare supervizată. Utilizează multiplicatori Lagrange pentru a găsi un hiperplan optim care să separe eficient clasele într-un set de date. Abilitatea AG de a depăși minime locale în spațiul parametrilor face ca optimizarea SVM să fie mai robustă și să obțină soluții mai consistente. Acesta este modul în care AG poate fi integrat cu SVM pentru a optimiza acești multiplicatori Lagrange și pentru a atinge o mai mare eficiență în rezolvarea problemelor de clasificare și regresie [2].



3. Modalitatea de rezolvare

Am implementat un SVM care să poată să facă distincția între două clase de cancer, malign și bening. Inițial se citesc instanțele despre setul de date, apoi generându-se aleatoriu valorile pentru genele cromozomilor necesari algoritmului evolutiv. Valorile sunt în intervalul [0;C], valoarea lui C fiind demonstrată în capitolul 6. Algoritmul evolutiv va parcurge pașii menționați în capitolul anterior timp de 100 de generații, în care va încerca să maximizeze și să ajusteze valorile multiplicatorilor Lagrange. În urma efectuării acestuia, se vor calcula vectorii suport, biasul, vectorul hiperplanului, margimea m și se va verifica că suma de alfa și clasă este 0.

4. Codul sursă și explicații aferente

Metoda Adjustment [2] - scopul principal al metodei Adjustment este de a asigura că suma multiplicatorilor Lagrange pentru clasele pozitive și negative este echilibrată în cadrul populației de cromozomi. În contextul problemei de clasificare a cancerului mamar, aceasta asigură că modelul SVM obținut din populație nu este sesizabil înclinat în favoarea uneia dintre clasele de date (clasa pozitivă sau negativă). Acest pas de ajustare este crucial pentru a evita convergența către soluții în care una dintre clase primește o atenție semnificativ mai mare, ceea ce ar putea duce la performanțe slabe ale modelului.

```
/// <summary>
/// Ajustează populația de cromozomi pentru a se asigura că sumele clasificatorilor de multiplicatori pentru clasele pozitive și negative
/// sunt echilibrate. Această metodă corectează valoarea genelor pentru a asigura conformitatea cu constrângerile problemei.
/// Este crucială pentru menținerea validității soluțiilor în contextul problemei de clasificare.
/// </summary>
/// <param name="chromosome">Chrommozomul care trebuie ajustată.</param>

2 references
public void Adjustment(Chromosome alpha)
{
    const int MaxIterations = 1000;
    int iterationCount = 0;

    double difference;
    do
    {
        double positiveSum = CalculateSum(alpha, 1);
        double negativeSum = CalculateSum(alpha, -1);
         difference = positiveSum - negativeSum;

        int geneIndex = SelectGeneIndex(alpha, positiveSum, negativeSum);

        UpdateGeneValue(alpha, geneIndex, Math.Abs(difference));

        positiveSum = CalculateSum(alpha, 1);
        negativeSum = CalculateSum(alpha, -1);
        difference = positiveSum - negativeSum;

        iterationCount++;

    } while (Math.Abs(difference) > 1e-6 && iterationCount < MaxIterations);
}
```

Metoda ComputeFitness [2] - calculează valoarea de fitness a unui cromozom specific în cadrul populației curente de cromozomi. Folosind o formulă specifică de fitness, această metodă evaluează cât de bine se potrivește cromozomul în contextul problemei de clasificare a cancerului mamar.

```
/// <summary>
/// Calculează fitness-ul unui cromozom în contextul întregii populații, conform metodologiei descrise în documentul de suport.
/// </summary>
/// <param name="chromosome">Cromozomul pentru care se calculează fitnessul.</param>
/// <param name="population">Populația întreagă.</param>
3 references
public void ComputeFitness(Chromosome chromosome, Chromosome[] population)
{
    double sum1 = 0.0;

    // Calculează primul termen din formula fitness-ului.
    // Acesta reprezintă suma valorilor genelor din populație.
    for (int i = 1; i < population.Length; ++i)
    {
        for (int j = 0; j < population[i].Genes.Length; ++j)
        {
            sum1 += population[i].Genes[j];
        }
    }

    // Calculează al doilea termen din formula fitness-ului, care este o sumă de produse.
    double sum21 = 0.0;
    for (int j = 1; j < population.Length; ++j)
    {
        double sum22 = 0.0;

        // Parcurge populația și calculează produsul scalar între genele cromozomilor.
        for (int i = 1; i < population.Length; ++i)
        {
            var x = population[i].Genes;
            var y = population[j].Genes;
            var scp = x.Zip(y, (xi, yi) => xi * yi).Sum(); // Produsul scalar dintre gene.

            // Adaugă produsul scalar, înmulțit cu clasele și valoarea returnată de funcția kernel.
            sum22 += population[i].Genes[0] * population[j].Genes[0] * population[i].Clasa * population[j].Clasa * Kernel(population[i].Instances, population[j].Instances);
        }

        sum21 += sum22;
    }

    // Calculează valoarea fitness-ului conform formulei date.
    chromosome.Fitness = -sum1 + 0.5 * sum21;
}
```

Metoda Kernel - implementează funcția de kernel RBF [3], utilizată în calculul fitness-ului cromozomilor. Această funcție măsoară similaritatea în spațiul de caracteristici, esențială pentru determinarea acurateței unui cromozom în rezolvarea clasificării cancerului mamar. Acest kernel, prin sensibilitatea sa la distanța dintre instanțe, ajută la definirea hiperplanului de separare în SVM.

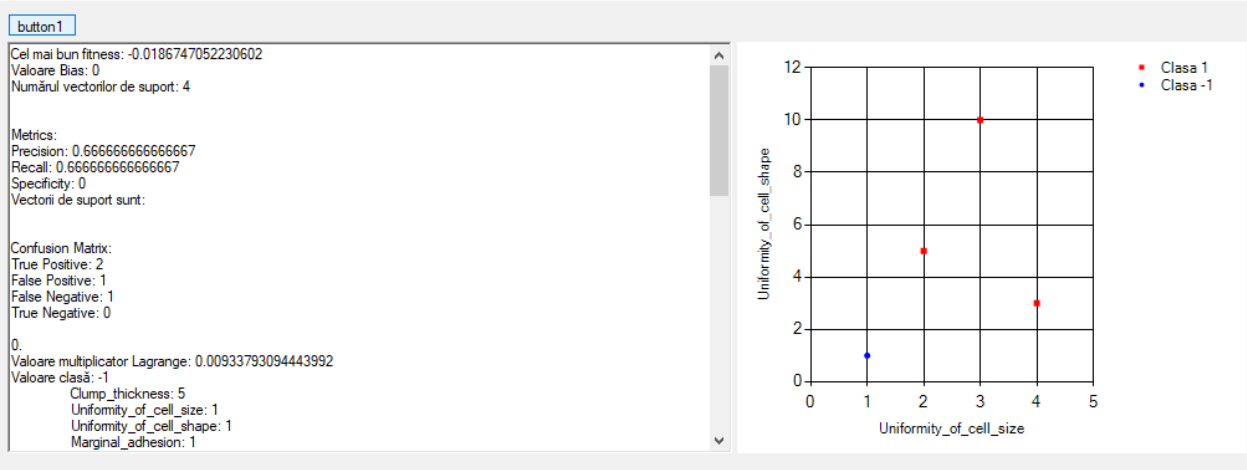
```
/// <summary>
/// Implementează funcția de kernel RBF (Radial Basis Function) cunoscută și sub numele de Gaussian,
/// utilizată pentru a măsura similaritatea în spațiul caracteristicilor în SVM.
/// </summary>
/// <param name="x">Primul vector de instanțe.</param>
/// <param name="y">Al doilea vector de instanțe.</param>
/// <returns>Valoarea calculată a kernelului RBF.</returns>
2 references
private double Kernel(double[] x, double[] y)
{
    // Calculează suma pătratelor diferențelor între elementele vectorilor,
    // ceea ce reprezintă pătratul distanței euclidiene.
    double sumOfSquares = 0;
    for (int i = 0; i < x.Length; i++)
    {
        sumOfSquares += Math.Pow(x[i] - y[i], 2);
    }

    // Aplică transformarea RBF pe distanța calculată, ajustată de parametrul gamma.
    // Rezultatul este o măsură a similarității: cu cât doi vectori sunt mai apropiați,
    // cu atât valoarea kernelului este mai mare.
    return Math.Exp(-Constants.Gamma * Math.Sqrt(sumOfSquares));
}
```

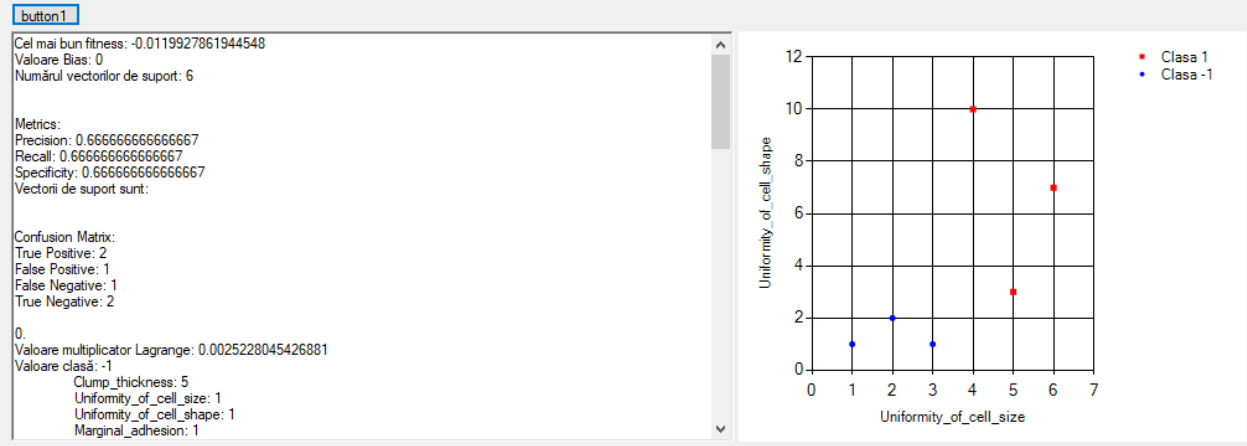
Metoda Solve - implementează logica principală a algoritmului genetic, preluată din [5]. Ea începe prin inițializarea populației de cromozomi, calculează fitness-ul fiecărui cromozom, ajustează populația pentru a satisface cerințele problemei și efectuează selecția, încrucișarea și mutația în fiecare generație. La final, returnează populația curentă, care conține soluții potențiale pentru problema de optimizare.

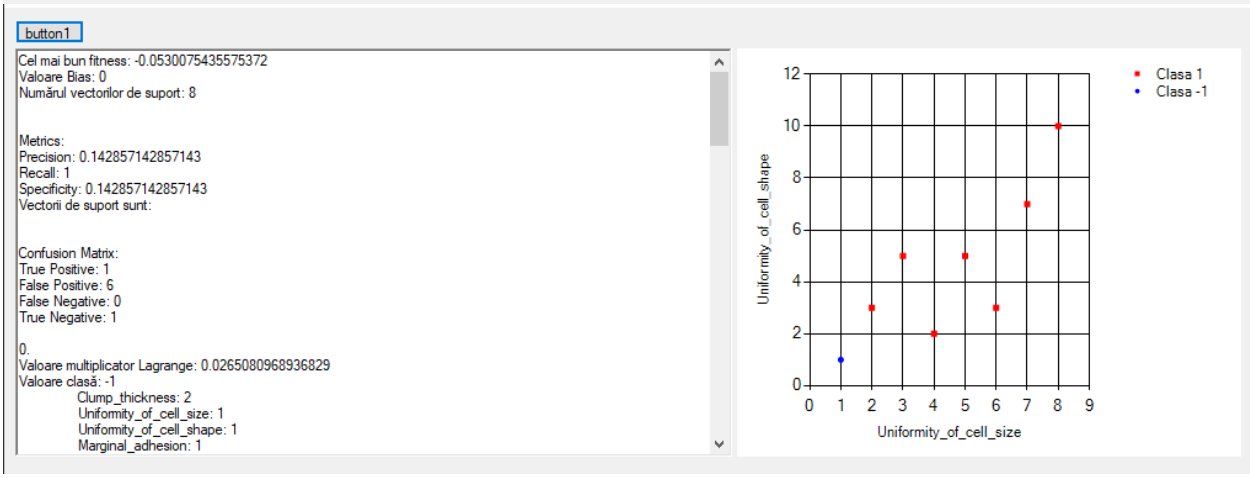
```
public Chromosome[] Solve()
{
    // Inițializează populația de cromozomi pentru algoritmul genetic
    Chromosome[] population = _breastCancer.PopulateChromosome();

    // Calculează fitness-ul fiecărui cromozom din populație
    for (int i = 0; i < population.Length; ++i)
    {
        _breastCancer.ComputeFitness(population[i], _trainingDataset);
    }

    // Ajustează populația conform anumitor criterii.
    for (int i=0; i < population.Length; i++)
    {
        Adjustment(population[i]);
    }

    // Execută algoritmul genetic pentru un număr definit de generații
    for (int generatie = 0; generatie < Constants.MaxGenerations; ++generatie)
    {
        // Inițializează noua populație
        Chromosome[] newPopulation = new Chromosome[population.Length];

        // Pastrează cel mai bun cromozom prin elitism
        newPopulation[0] = GetBest(population);

        // Generează restul noii populații
        for (int i = 1; i < population.Length; ++i)
        {
            // Selectează doi părinți
            Chromosome mother = Selection_Tournament(population);
            Chromosome father = Selection_Tournament(population);

            // Aplică încrucișarea
            Chromosome child = Crossover_Arithmetic(mother, father, Constants.CrossoverRate);

            // Aplică mutația
            Mutation_Reset(child, Constants.MutationRate);

            // Ajustează noul copil
            Adjustment(child);

            // Calculează fitness-ul noului cromozom
            _breastCancer.ComputeFitness(child, _trainingDataset);

            // Adaugă copilul în noua populație
            newPopulation[i] = child;
        }

        // Înlocuiește populația veche cu cea nouă
        population = newPopulation;

        // Afișează informații despre starea curentă a algoritmului
        Console.WriteLine("Generatia: " + (generatie + 1));
        Console.WriteLine("Fitnes: " + population[0].Fitness);
        Console.WriteLine("Alfa: " + population[0].Genes[0]);
    }

    // Returnează populația finală.
    return population;
```

5. Rezultatele obținute

Rezultatele curente sunt provenite în urma următorilor parametri setați: 100 de instanțe pe setul de date, 50 de indivizi într-o populație, 100 de populații, C=0.1, Gamma=0.001. Acești parametri au fost aleși în urma repetatelor teste.







Interpretarea rezultatelor:

• ***Performanța Generală***:
  – Modelele obținute au un fitness relativ scăzut (apropiat de 0), indicând că optimizarea se concentrează pe identificarea vectorilor de suport cu greutăți semnificative.
  – Performanța în general nu este foarte ridicată, sugerând că poate exista spațiu pentru îmbunătățiri ale algoritmului sau ajustări ale parametrilor.

• ***Numărul de Vectori de Suport***:
  – Prezența falselor pozitive în matricea de confuzie indică o anumită incertitudine în identificarea cazurilor negative și poate sugera o sensibilitate crescută la anumite instanțe.
  – Precision, Recall și Specificity: Precision-ul este relativ scăzut, ceea ce indică o precizie redusă în identificarea cazurilor pozitive. Recall-ul variază, sugerând o abilitate variabilă de a identifica cazurile pozitive. Specificitatea este destul de scăzută, indicând o eficiență redusă în evitarea falselor pozitive.

• ***Recomandări pentru îmbunătățirea acestui proiect***:
  – Ajustările suplimentare ale parametrilor algoritmului genetic pot fi explore pentru a îmbunătăți convergența și performanța.
  – Analiza detaliată a instanțelor individuale și a vectorilor de suport poate oferi înțelegeri specifice asupra influenței acestora asupra performanței modelului.

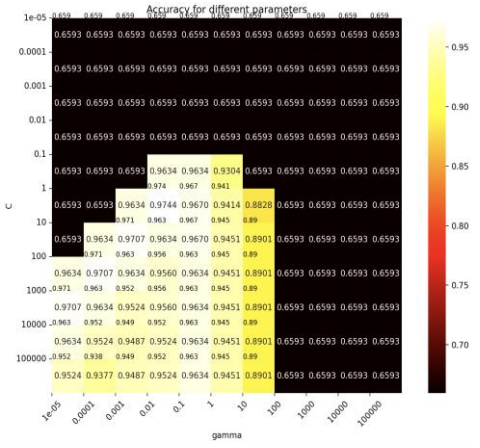
De asemenea, am analizat și evoluția marginii. Spre exemplu, în cadrul unei rulări obținem un astfel de rezultat:

```
Marginea este: 1.31375360868411
```

Marginea reprezintă distanța dintre hiperplanul de decizie și cel mai apropiat vector de suport. O valoare mai mare a marginii indică o separare mai eficientă între clase în spațiul caracteristic.

6. Manipularea Setului de Date și Alegerea Parametrilor

Am dezvoltat un program Python care automatizează optimizarea unei mașini cu vectori suport (SVM) folosind kernelul radial (RBF). Parametrii critici, C și gamma, sunt esențiali pentru performanța SVM, iar alegerea incorectă a acestora poate duce la sub-antrenare sau suprarezolvare. Parametrul C reglează compromisul între lărgimea marginii de decizie și clasificarea corectă, iar gamma controlează influența locală a punctelor de antrenareProgramul utilizează un algoritm evolutiv pentru a găsi combinații optime de C și gamma, asigurând astfel o optimizare eficientă a SVM pentru generalizare și performanță ridicate. Rezultate obținute pe setul de date din [1] cu sursa programului din [4]:




```
** Best parameters found during Grid Search are {'C': 1, 'gamma': 0.01}
** Cross-validated Accuracy: 0.9743589743589745
```

Sesizare dupa aplicarea practia a valorilor în proiectul creat manual:

- Chiar dacă, în etapa de selecție a celor mai buni parametri, valoarea optimă a parametrului C a fost identificată ca fiind 1, practica a relevat că modelul SVM performează mai bine când C este setat la 0.1. Această discrepanță poate fi explicată prin sensibilitatea algoritmului la datele specifice de antrenare utilizate în acel moment, dimensiunea setului de date și distribuția specifică a acestora.

7. Concluzii

Proiectul a reprezentat o încercare de a optimiza diagnosticul cancerului mamar prin utilizarea unui algoritm evolutiv în cadrul Support Vector Machine (SVM). Cu toate acestea, rezultatele actuale indică o performanță generală modestă. Este evident că există loc pentru îmbunătățiri și ajustări ale algoritmului sau parametrilor. Recomandările pentru viitor includ explorarea suplimentară a ajustărilor de parametri și analiza detaliată a vectorilor de suport și instanțelor individuale pentru a obține o înțelegere mai profundă a influenței acestora.

8. Rolul membrilor din echipă

Hușman Carla Gabriela:

- Implementarea codului, documentație, interfață

Pașa Anamaria-Larisa:

- Implementarea codului, documentație, program Python

Pîslariu Andreea:

- Algoritm evolutiv, documentație, documentarea codului

9. Bibliografie

[1] Dr. Wolberg, “Breast Cancer Wisconsin (Original)”, [Online]. Available: https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original
[2] Madson Luiz Dantas Dias and Ajalmar R. Rocha Neto, “Evolutionary Support Vector Machines: A Dual Approach”, 2016, DOI:10.1109/CEC.2016.7744058 [Online], Available: https://www.researchgate.net/publication/309565420_Evolutionary_Support_Vector_Machines_A_Dual_Approach
[3] Curs 12 Inteligență Artificială - Florin Leon
[4] Ajitesh Kumar, “SVM RBF Kernel Parameters” [Online], Available: https://vitalflux.com/svm-rbf-kernel-parameters-code-sample/
[5] Curs 04 Inteligență Artificiala - Florin Leon