

Universitatea Tehnica „Gheorghe Asachii” din Iasi

Facultatea de Automatica si Calculatoare

Domeniul Calculatoare si Tehnologia Informatiei

Specializarea Tehnologia Informatiei

Proiect Inteligenta Artificiala

- Antrenarea unei retele neuronale utilizand algoritmi evolutivi -

Nistor Florin 1408B

An universitar 2023-2024

Cuprins

Cuprins.....	2
1. Capitolul 1: Tema Proiectului.....	3
1.1.Enunțarea temei proiectului.....	3
1.2.Motivul alegerii temei.....	4
2. Capitolul 2. Arhitectura generală.....	5
2.1.Componente principale.....	5
2.2.Relatii între componente.....	5
2.3.Fluxul de date.....	5
3. Capitolul 3. Funcționalitatea.....	6
3.1.Descriere interacțiune.....	6
3.2.Controale.....	7
4. Capitolul 4. Rolul fiecărui membru al echipei.....	9
5. Capitolul 5. Rezultate Algoritm Evolutiv.....	9
6. Capitolul 6. Teste automate pentru a demonstra buna funcționare a programului.....	11
6.1.Modulul de Testare unitară.....	12
6.2.Rezultate.....	12
7. Capitolul 7. Explicații suplimentare.....	13
7.1.Utilizarea algoritmului evolutiv.....	13
7.2.Utilizarea rețelei neuronale.....	14
8. Capitolul 8. Bibliografie.....	14

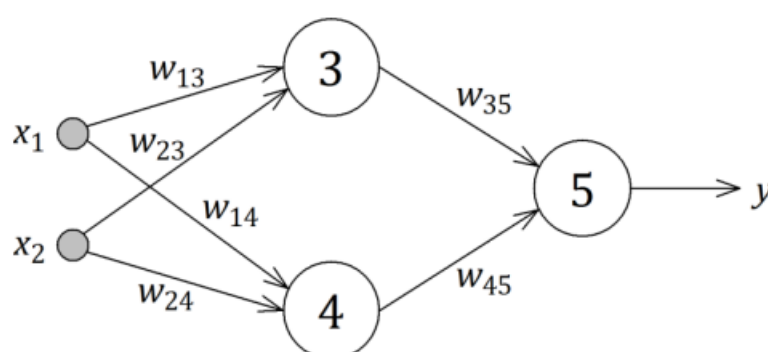
1. Capitolul 1.

Tema Proiectului:

1.1. Enunțarea temei proiectului:

Antrenarea unei rețele neuronale de tip perceptron multistrat cu o structură predefinită (un număr specificat de straturi ascunse și neuroni) cu ajutorul unui algoritm evolutiv.

Se dă un prototip de aplicație pentru implementarea unui perceptron cu un singur strat și a unui perceptron cu un strat ascuns, având configurația din figura următoare:



Prototipul include interfața grafică cu utilizatorul și desenarea regiunilor de decizie. Întrucât aici se presupune că ieșirile neuronilor aparțin intervalului $[0, 1]$, se folosesc următoarele expresii pentru funcțiile de activare:

- funcția prag:

$$f(s) = \begin{cases} 0, & s < 0 \\ 1, & s \geq 0 \end{cases}$$

- funcția semiliniară:

$$f(s) = \begin{cases} 0, & s \leq 0 \\ s, & s \in (0, 1) \\ 1, & s \geq 1 \end{cases}$$

- funcția sigmoidă unipolară:

$$f(s) = \frac{1}{1 + e^{-s}}$$

Se dorește determinarea ponderilor conexiunilor dintre neuroni și a valorilor prag astfel încât rețeaua să aproximeze câteva funcții binare elementare. De exemplu:

Funcția de activare	Funcția de aproximat	w_{13}	w_{23}	w_{14}	w_{24}	w_{35}	w_{45}	θ_3	θ_4	θ_5
prag	nand	1	0	0	1	-0.5	-0.5	0.5	0.5	-0.5
semiliniară	or	1	1	1	1	0.5	0.5	0	0	0
sigmoidă	and	0	-2	-6	-6	0	-5	0	-9	-3
sigmoidă	or	-5	-5	6	6	-3	3	-3	3	0
sigmoidă	xor	-7	-7	-9	-9	5	-5	-10	-5	2

1.2. Motivul alegerii temei:

Am ales sa implementez algoritmul evolutiv pentru optimizarea rețelei neuronale pentru modul in care algoritmi evolutivi pot contribui la imbunatatirea performantei rețelei neuronale. De asemenea, am vazut potentialul acestui tip de abordare in rezolvarea problemelor complexe si am dorit sa explorez aceasta directie in proiectul meu.

2. Capitolul 2.

Arhitectura generală:

2.1. Componente principale:

- **Algoritmul Evolutiv:** Aceasta este componenta centrala a proiectului si include implementarea selectiei, crossover-ului si mutatiei, precum si logica pentru gestionarea populatiei de cromozomi.
- **Reteaua Neurala:** Acest modul include implementarea rețelei neurale si metodele necesare pentru actualizarea ponderilor in functie de cromozomi si evaluarea performantei rețelei.
- **Interfata Grafica:** Acest modul include o simpla interfata ce permite utilizatorului un control asupra modificarii variabilelor, generarea de valori prin antrenare, printr-un singur strat sau multistrat si un buton pentru optimizarea valorilor, utilizand algoritmul evolutiv.

2.2. Relatii intre componente:

- **Algoritm Evolutiv si Reteaua Neurala:** Exista o relatie stransa intre algoritmul evolutiv si reseaua neurala. Algoritmul evolutiv gestioneaza generarea, selectia si actualizarea populatiei de cromozomi, iar reseaua neurala utilizeaza acesti cromozomi pentru a ajusta ponderile si pentru evaluarea performantei.

2.3. Fluxul de date:

- **Generarea initiala:** Algoritmul evolutiv genereaza initial o populatie de cromozomi, utilizand metoda „MakeChromosome” din interfata „IoptimizationProblem”.
- **Evaluarea Performantei:** Reteaua neurala este utilizata pentru a evalua performanta fiecarui cromozom in populatie prin intermediul metodei „ComputeFitness” din interfata „IoptimizationProblem”.
- **Actualizarea Ponderilor:** Reteaua neurala ajusteaza ponderile in functie de cromozomii evaluati.
- **Crossover si Mutatie:** Algoritmul evolutiv utilizeaza operatiile de crossover si mutatie pentru a crea noi cromozomi in fiecare generatie.

3. Capitolul 3.

Funcționalitatea:

3.1. Descriere interacțiune:

Interfata grafica initiaza procesul prin apelul metodei „MakeChromosome” din interfata „IoptimizationProblem” pentru a genera cromozomii initiali. Algoritmul evolutiv preia acesti cromozomi si ii evalueaza prin apelul metodei „ComputeFitness” pentru a calcula performanta.

Algoritmul evolutiv utilizeaza metodele de selectie, crossover si mutatie pentru a genera o noua populatie de cromozomi.

Pentru fiecare cromozom nou creat, metoda „ComputeFitness” este apelata pentru evaluarea performantei acestuia.

Dupa evaluarea performantei, reseaua neurala este actualizata cu noile ponderi calculate pe baza cromozomilor din populatie.

Metoda „UpdateNetworkWeights” din interfata „IoptimizationProblem” este apelata pentru a actualiza ponderile retelei cu cele ale cromozomului curent.

Procesul de optimizare continua pentru un numar specific de generatii. La finalul procesului, rezultatele, cum ar fi cromozomul optimizat sau performanta finala, vor fi afisate in interfata grafica.

3.2. Controale:

Retele neuronale - Regiuni de decizie

Perceptron cu un singur strat

Funcție de activare prag

Perceptron cu un strat

w13 1

w23 1

t3 1.5

Perceptron multistrat

w13 -7 w35 5

w23 -7 w45 -5

w14 -9 t5 2

w24 -9

t3 -10

t4 -5

Antrenare

00	0
01	0
10	0
11	1

Optimizare Utilizand Algoritm Evolutiv

Antreneaza SLP Antreneaza MLP

Deseneaza Despre Iesire

Controalele sunt pe baza de butoane descriptive in cadrul interfetei grafice utilizator. Aplicatia este intuitiva si usor de folosit pentru oricine doreste utilizarea acesteia.

Perceptron cu un singur strat

Perceptron cu un singur strat

Perceptron multistrat

Putem alege daca reteaua neurala sa fie cu un Perceptron cu un singur strat sau multistrat.

Funcție de activare prag

Funcție de activare prag

Funcție de activare semiliniara

Funcție de activare sigmoida unipolara

Apoi putem alege ce functie dorim ca algoritmul nostru sa utilizeze.

Perceptron cu un strat

w13

w23

t3

Putem seta manual cu ce valori dorim noi sau putem sa Antrenam algoritmul sa ne genereze el valori corecte pentru noi. Selectand ce rezultat ne dorim sa avem.

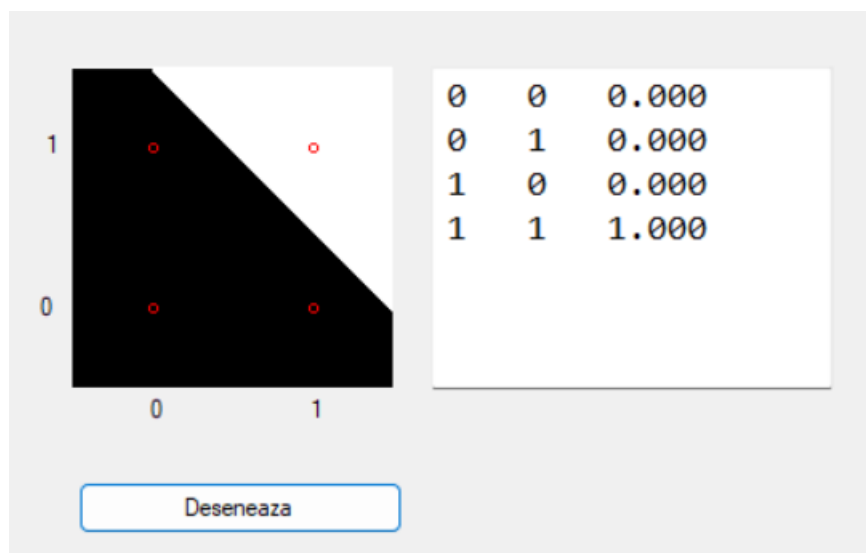
Antrenare

0 0	<input type="text" value="0"/>
0 1	<input type="text" value="0"/>
1 0	<input type="text" value="0"/>
1 1	<input type="text" value="1"/>

Optimizare Utilizand Algoritm Evolutiv

Antreneaza SLP Antreneaza MLP

In cazul in care rezultatul nu este intocmai perfect sau ne dorim o estimare cat mai apropiata de solutia dorita putem sa optimizam valorile utilizand Algoritmul Evolutiv. Astfel putand avea valori ce sa ne ofere un rezultat cat mai exact.



Intr-un final se va afisa atat solutia pe care ne-o doream cat si un grafic al solutiei.

4. Capitolul 4.

Rolul fiecărui membru al echipei:

- Implementare algoritm evolutiv pe o retea neuronală – Nistor Florin
- Documentatie – Nistor Florin

5. Rezultate algoritm evolutiv:

Deoarece algoritmul evolutiv are nevoie de o populatie initiala si un numar de generatii cat mai mare pentru a avea o solutie cat mai buna am decis sa rulez de acasa cateva exemple avand ca parametrii:

populationSize = 100

maxGenerations = 20

crossoverRate = 0.4

mutationRate = 0.33

MLP:

```
textBoxWw35.Text = w[6].ToString("F6", CultureInfo);
textBoxWw45.Text = w[7].ToString("F6", CultureInfo);
textBoxWt5.Text = w[8].ToString("F6", CultureInfo);

//Ce am adaugat pentru algoritmul evolutiv
private void buttonOptimize_Click(object sender, EventArgs e)
{
    // Apelul algoritmului evolutiv pentru a optimiza rețeaua
    int populationSize = 100; // Setează dimensiunea populației
    int maxGenerations = 20; // Setează numărul maxim de generații
    double crossoverRate = 0.4; // Setează rata de crossover
    double mutationRate = 0.33; // Setează rata de mutație

    Chromosome optimizedChromosome = evolutionaryAlgorithm.Optimize(populationSize, maxGenerations, crossoverRate, mutationRate);

    // Actualizează ponderile rețelei neuronale cu cele optimizate
    UpdateNetworkWeights(optimizedChromosome);

    // <summary>
    // Selectează tipul de perceptron atunci când utilizatorul alege
    // </summary>
    private void comboBoxType_SelectedIndexChanged(object sender, EventArgs e)
    {
        // ...
    }
}
```

Nu avem o solutie perfecta dar daca marim numarul de generatii o sa avem sanse mai mari sa dam peste solutia perfecta

SLP:

```

//Ce am adaugat pentru algoritmul evolutiv
private void buttonOptimize_Click(object sender, EventArgs e)
{
    // Apelul algoritmului evolutiv pentru a optimiza rețeaua
    int populationSize = 100; // Setează dimensiunea populației
    int maxGenerations = 20; // Setează numărul maxim de generații
    double crossoverRate = 0.4; // Setează rata de crossover
    double mutationRate = 0.33; // Setează rata de mutație

    Chromosome optimizedChromosome = evolutionaryAlgorithm.Solve(
        // Actualizează ponderile rețelei neuronale cu cele ale cromozomului optimizat
        UpdateNetworkWeights(optimizedChromosome);

    // <summary>
    // Selectează tipul de perceptron atunci când utilizatorul a selectat un tip
    // </summary>
    reference
    private void comboBoxType_SelectedIndexChanged(object sender, EventArgs e)
    {
        // ...
    }
}

```

Perceptron cu un singur strat

Perceptron cu un strat

w13: 0.2
w23: 0.1
t3: 0.3

Funcție de activare prag

Perceptron multistrat

w13: -7, w23: -7, w14: -9, w24: -9, t3: -10, t4: -5, w35: 5, w45: -5, t5: 2

Antrenare

0 0: 0, 0 1: 0, 1 0: 0, 1 1: 1

Optimizează Utilizând Algoritm Evolutiv

Antrenează SLP, Antrenează MLP

Desenează, Despre, Iesire

```

//Ce am adaugat pentru algoritmul evolutiv
private void buttonOptimize_Click(object sender, EventArgs e)
{
    // Apelul algoritmului evolutiv pentru a optimiza rețeaua
    int populationSize = 100; // Setează dimensiunea populației
    int maxGenerations = 20; // Setează numărul maxim de generații
    double crossoverRate = 0.4; // Setează rata de crossover
    double mutationRate = 0.33; // Setează rata de mutație

    Chromosome optimizedChromosome = evolutionaryAlgorithm.Solve(
        // Actualizează ponderile rețelei neuronale cu cele ale cromozomului optimizat
        UpdateNetworkWeights(optimizedChromosome);

    // <summary>
    // Selectează tipul de perceptron atunci când utilizatorul a selectat un tip
    // </summary>
    reference
    private void comboBoxType_SelectedIndexChanged(object sender, EventArgs e)
    {
        // ...
    }
}

```

Perceptron cu un singur strat

Perceptron cu un strat

w13: 0.2, w23: 0.1, t3: 0.3

Funcție de activare semiliniară

Perceptron multistrat

w13: -7, w23: -7, w14: -9, w24: -9, t3: -10, t4: -5, w35: 5, w45: -5, t5: 2

Antrenare

0 0: 0, 0 1: 0, 1 0: 0, 1 1: 1

Optimizează Utilizând Algoritm Evolutiv

Antrenează SLP, Antrenează MLP

Desenează, Despre, Iesire

```

//Ce am adaugat pentru algoritmul evolutiv
private void buttonOptimize_Click(object sender, EventArgs e)
{
    // Apelul algoritmului evolutiv pentru a optimiza rețeaua
    int populationSize = 100; // Setează dimensiunea populației
    int maxGenerations = 20; // Setează numărul maxim de generații
    double crossoverRate = 0.4; // Setează rata de crossover
    double mutationRate = 0.33; // Setează rata de mutație

    Chromosome optimizedChromosome = evolutionaryAlgorithm.Solve(
        // Actualizează ponderile rețelei neuronale cu cele ale cromozomului optimizat
        UpdateNetworkWeights(optimizedChromosome);

    // <summary>
    // Selectează tipul de perceptron atunci când utilizatorul a selectat un tip
    // </summary>
    reference
    private void comboBoxType_SelectedIndexChanged(object sender, EventArgs e)
    {
        // ...
    }
}

```

Perceptron cu un singur strat

Perceptron cu un strat

w13: 0.1, w23: -0.4, t3: 0.1

Funcție de activare prag

Perceptron multistrat

w13: -7, w23: -7, w14: -9, w24: -9, t3: -10, t4: -5, w35: 5, w45: -5, t5: 2

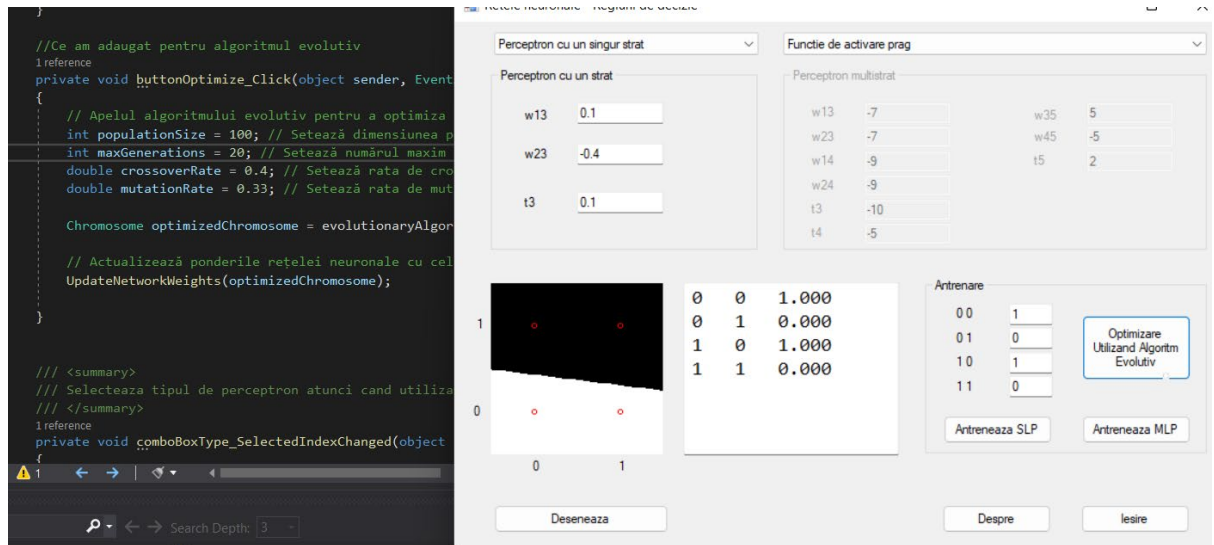
Antrenare

0 0: 0, 0 1: 0, 1 0: 1, 1 1: 0

Optimizează Utilizând Algoritm Evolutiv

Antrenează SLP, Antrenează MLP

Desenează, Despre, Iesire



6. Capitolul 6.

Teste automate pentru a demonstra buna funcționare a programului:

6.1. Modulul de Testare unitară:

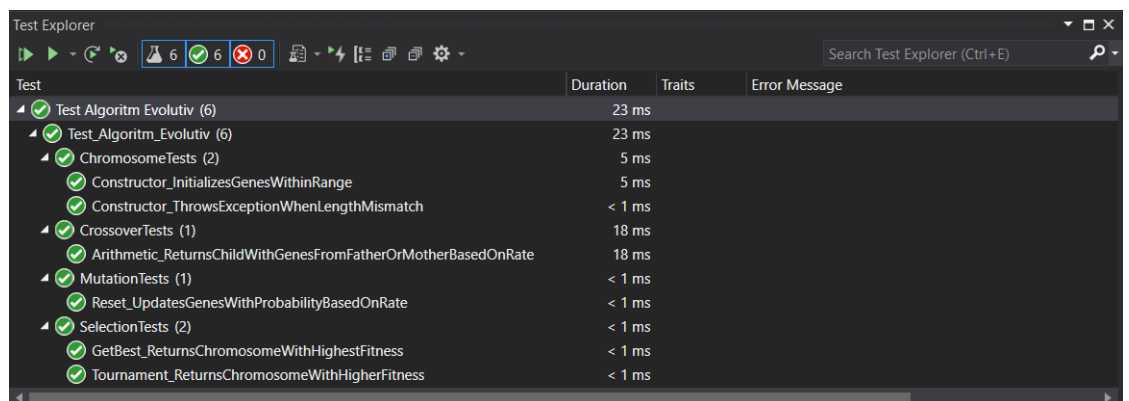
Modulul de testare unitară conține o serie de clase de testare, fiecare responsabilă pentru o funcționalitate anume.

- **Constructor_InitializeGenesWithinRange:** Acest test verifică corectitudinea initializării cromozomului în ceea ce privește valorile genelor. Se asigură că gena fiecărui cromozom este initializată la o valoare cuprinsă între limitele specificate (',minValues' și ',maxValues)'). Testul este conceput pentru a se asigura că cromozomul este creat cu genele într-un interval valid.
- **Constructor_ThrowsExceptionWhenLengthMismatch:** Verifică dacă constructorul cromozomului aruncă o excepție atunci când lungimile vectorilor ',minValues' și ',MaxValues' nu sunt aceleași cu ',noGenes'. Scopul acestui test este de a confirma că se lansează o excepție în cazul în care datele de intrare nu sunt consistente.
- **Arithmetic_ReturnsChildWithGenesFromFatherOrMotherBasedOnRate:** Verifică funcționalitatea corectă a

crossover-ului aritmetic. Se asigura ca copilul generat are gene preluate de la mama sau tata in functie de rata specificata. Testul este conceput pentru a valida daca crossover-ul aritmetic functioneaza asa cum este asteptat.

- **Reset_UpdatesGenesWithProbabilityBasedOnRate:**
Acest test valideaza corectitudinea functionarii mutatiei de resetare. Se asigura ca gena unui copil este actualizata cu probabilitatea specificata, iar noua valoare este in intervalul permis. Scopul este de a confirma ca mutatia de resetare are loc doar cu probabilitatea specificata.
- **GetBest_ReturnsChromosomeWithHighestFitness:**
Verifica daca metoda ,GetBest' returneaza cromozomul cu cea mai mare valoare de fitness dintr-o populatie data. Acest test confirma corectitudinea functionarii algoritmului de selectie a celui mai bun cromozom.
- **Tournament_ReturnsChromosomeWithHigherFitness:**
Acest test asigura functionarea corecta a selectiei turneu. Verifica daca, intr-un turneu, este selectat cromozomul cu cea mai mare valoare de fitness. Scopul testului este de a valida ca selectia turneu alege cromozomul cu cea mai mare valoare de fitness.

6.2. Rezultate:



Test	Duration	Traits	Error Message
Test Algorithm Evolutiv (6)	23 ms		
Test_Algorithm_Evolutiv (6)	23 ms		
ChromosomeTests (2)	5 ms		
Constructor_InitializesGenesWithinRange	5 ms		
Constructor_ThrowsExceptionWhenLengthMismatch	< 1 ms		
CrossoverTests (1)	18 ms		
Arithmetic_ReturnsChildWithGenesFromFatherOrMotherBasedOnRate	18 ms		
MutationTests (1)	< 1 ms		
Reset_UpdatesGenesWithProbabilityBasedOnRate	< 1 ms		
SelectionTests (2)	< 1 ms		
GetBest_ReturnsChromosomeWithHighestFitness	< 1 ms		
Tournament_ReturnsChromosomeWithHigherFitness	< 1 ms		

7. Capitolul 7.

Explicații suplimentare:

7.1. Utilizarea algoritmului evolutiv:

Algoritmul evolutiv este o paradigma de optimizare inspirată de procesele naturale de evoluție. În contextul proiectului, acesta este folosit pentru a optimiza ponderile rețelei neuronale.

- **Populație:** Algoritmul începe cu o populație de cromozomi, fiecare reprezentând un set diferit de ponderi pentru rețeaua neurală.
- **Evaluare:** Fiecare cromozom din populație este evaluat prin intermediul rețelei neuronale pentru a măsura performanța. Acesta este un pas important pentru a determina calitatea fiecărui individ din populație.
- **Selectia:** Cromozomii sunt selecționați pentru a forma o nouă generație, cu o probabilitate mai mare pentru cei cu performanțe mai bune. Se utilizează diferite metode de selecție, cum ar fi turneul sau selecția elitistă.
- **Crossover:** Peste perechile de cromozomi selecționați, se aplică crossover (încrucișare) pentru a genera cromozomi copii. Acest proces combină caracteristicile alese de la ambii părinți și creează indivizi cu noi caracteristici.
- **Mutație:** Se aplică mutații cu o anumită probabilitate pentru a introduce variabilitatea în populație. Mutațiile pot consta în modificarea aleatorie a genelor cromozomilor.
- **Generații:** Procesul de selecție, crossover și mutație se repetă pe de-a lungul a mai multe generații pentru a permite evoluția populației către soluții mai bune.

7.2. Utilizarea rețelei neuronale:

Rețeaua neurală este folosită pentru a realiza o anumită sarcină, în cazul nostru, pentru a face predicții sau clasificări bazate pe datele de intrare.

- **Arhitectura rețelei:** Descrie cum sunt organizate straturile și conexiunile neuronale în rețeaua noastră. Acesta poate include straturi de intrare, straturi ascunse și straturi de ieșire.
- **Antrenarea:** Ponderile inițiale ale rețelei sunt optimizate pentru a produce ieșiri cât mai apropiate posibil de ieșirile așteptate. Acest proces implică utilizarea unui set de date de antrenare și un algoritm de optimizare.

- Fine-tuning: In cadrul algoritmului evolutiv, ponderile rețelei sunt ajustate iterativ pentru a îmbunătăți performanța generală a rețelei.

8. Capitolul 8.

Bibliografie:

- **Inteligența Artificială – Laborator 12 – Rețele neuronale: regiuni de decizie de Florin Leon**
- **Inteligența Artificială – Laborator 8 – Algoritmi Evolutivi de Florin Leon**
- https://en.wikipedia.org/wiki/Evolutionary_algorithm
- https://ro.wikipedia.org/wiki/Re%C8%9Bea_neural%C4%83

