**UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI**

**FACULTATEA DE ŞTIINŢE APLICATE**

**Matematică şi Informatică Aplicată în Inginerie**

# PROIECT DE DIPLOMĂ

CONDUCĂTOR ŞTIINŢIFIC,
**Lect.univ.dr. Iuliana MUNTEANU**

ABSOLVENT,
**Soare Robert Daniel**

**Bucureşti**

**2021**

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI
FACULTATEA DE ŞTIINŢE APLICATE
Matematică şi Informatică Aplicată în Inginerie


**Aprobat Decan**,

**Prof.dr. Emil PETRESCU**

# PROIECT DE DIPLOMĂ

## Învătarea automată pentru un agent într-un mediu 2D

.....

CONDUCĂTOR ŞTIINŢIFIC,
**Lect.univ.dr. Iuliana MUNTEANU**

ABSOLVENT,
**Soare Robert Daniel**

**Bucureşti**

**2021**

# Cuprins

**Introducere** **4**

**1 Învăţare automată** **7**
1.1 Istoric . . . . . 7
1.2 Clasificare . . . . . 8
1.3 Industrie . . . . . 9
1.4 Programe software pentru dezvoltare . . . . . 10
1.5 Big Data . . . . . 10

**2 Reţele neuronale artificiale** **13**
2.1 Introducere . . . . . 13
2.2 Structură . . . . . 13
2.3 Funcţii de activare şi metode de optimizare . . . . . 15
2.4 Tensorflow . . . . . 16

**3 Metode de învăţare** **17**
3.1 Lanţ şi proces Markov . . . . . 17
3.2 Ecuaţia Bellman . . . . . 18
3.3 Q-Learning . . . . . 18

**4 Aplicaţie** **19**
4.1 Structură . . . . . 19
4.2 Simulator . . . . . 20
4.3 Interfaţă . . . . . 21
4.4 Agent . . . . . 21
4.5 Model de învăţare . . . . . 22

**Concluzii finale** **22**

**Bibliografie** **24**

**Index** **25**

# Introducere

Învăţarea automată a devenit un subiect de interes din ce în ce mai important, această fiind utilizată în vaste domenii, precum: industria auto, alimentară, agricolă, bancară, aerospaţială şi mai cu seamă în industria tehnologiei informaţiei. Unul din rolurile ei cele mai importante constă în analiza şi clasificarea datelor, predicţia unor evenimente în baza unor fapte deja întâmplate, crearea unui profil virtual pentru un grup de utilizatori, etc.

Datorită marei conectivităţi dintre oamenii din ziua de astăzi; sistemele politice, economice şi relaţiile interumane au devenit extrem de complexe. Totul a devenit interconectat. O ideea a unui singur individ poate fi transmisă pe tot globul pământesc, aceasta idee putand afectând milioane de oameni în diverse locuri şi a cărui impact politic şi economic poate fi greu de estimat. De asemenea, un incident economic local, un dezastru natural, sau un conflic politic dintre două ţări pot avea efecte devastatoare asupra economiei globale şi a structurii geopolitice curente.

Fiecare eveniment din ziua de astăzi are o influenţă mai mică sau mai mare aupra acestei mari reţele de sisteme ale civilizaţiei umane. Întrebarea naturală la această dilemna este: putem face o estimare aupra acestor evenimente şi ale cazurilor lor speciale? Se poate, şi asta datorită faptului ca multe evenimente sunt monitorizate şi înregistrate, precum: tranzacţiile bancare, documente legislative şi juridice, vremea, traseele şi destinaţiile maşinăriilor de transport marfă (automobile, avioane, vapoare), discursuri şi opinii în reţele sociale, date medicale din dispozitive inteligente (telefoane smart, ceasuri şi brătari smart), date provenite din simulări virtuale sau experimente.

Tot acest mare volum de informaţii şi metodele de manipulare, stocare întră în aşa numita categorie *Big Data*. Analiză acestui volum imens de date devine o sarcină foarte dificilă şi laborioasă în cazul metodelor convenţionale de analiză a datelor folosind statistică clasică. În esenţă, învăţarea automată se foloseste atăt de teoria clasică cât şi de noile descoperiri în calculul numeric pentru a crea modele matematice dinamice care pot acumula cunoţinţe şi acţiona în baza lor folosind toate datele pe care le primeşte ca set de învăţare.

În această lucrare se va analiza cum algoritmii de învăţare automată pot fi folosiţi în crearea unui agent autonom care să îndeplinească sarcinii într-un spaţiu 2-dimensional. Problema constă în rezolvarea unui traseu de tip matrice în care agentul trebuie să ajungă la destinaţie fără a produce un accident.

Analiza se va face cu ajutorul unei aplicaţii web interactive în care vom simula mediul

nostru 2-dimensional reprezentat de un labirint şi care ne va permite analiza datelor furnizate de către agent în timpul sesiuni de antrenament pentru determinarea eficienţei şi fiabilităţii algoritmilor.

În primul capitol este descris termenul de învăţare automată, care sunt subdomeniile sale, cum este folosit în îndustire.

În al doilea capitol este o mică introducere pentru reţelele neuronale artificiale.

În al treilea capitol vor fi prezentaţi căţiva algoritmi de învăţare, iar al patrulea descrierea aplicaţiei.

În al patrulea capitol se va descrie structura aplicaţiei web, a componentelor sale şi modul cum acestea interacţionează.

# Capitolul 1

# Învăţare automată

## 1.1 Istoric

Învăţarea automată este o ramură a înteligenţei artificiale care se ocupă cu studiul tehnicilor şi metodelor prin care se oferă unui calculator abilitatea de a învăţa. Prin învăţare ne referim la posibilitatea de a oferii o decizie în baza unor cunoştinţe deduse din experienţe anterioare.

Multe tehnici din învăţarea automată au la bază modelul de interacţiune al neuronilor, descris de către Donal Hebb în cartea sa *The Organization of Behavior* [1]. Termenul de învăţare automată (în engleză *machine learning*) a aparut în anul 1953, dat de Arthur Samuel, creatorul unui program de jucat checker, capabil să ia decizii bazate pe experienţele anterioare [2]. În anul 1957, Frank Rosenblatt crează Perceptron-ul - utilizat în crearea unui calculatorul capabil să recunoască forme într-o imagine - folosindu-se de observaţiile din lucrările lui Donald Hebb şi Arthur Samuel. Perceptron-ul de unul singur are o putere destul de limitată, dar odată cu descoperirea utilizării sale în combinaţii de mai multe straturi a dat naştere la termenul de reţea neuronală.

De-a lungul timpului, acest domeniu a avut o evoluţie înceată, un factor important find capabilităţile limitate de procesarea ale calculatoarelor. Dar odată cu avansurile tehnologice, cercetarea în acest domeniu a început să fie din ce în ce mai activă, în ultimii ani culminând cu evenimente care au atras interesului publicului general, precum: IBM's Deep Blue, IBM's Watson, Google's Deepmind şi Google's AlphaGo.

## 1.2 Clasificare

Fiind un domeniu foarte vast şi cuprinzţor, aceasta se împarte în 3 mari categorii:

- Învăţare supervizată
- Învăţare nesupervizată
- Învăţare prin recompensă

În învăţarea supervizată, procesul de antrenare se bazează pe analiza unor date formate din perechi de valori intrare-ieşire (set de date etichetat) pentru calibrarea funcţiilor de deducere. Este folosit pentru rezolvarea problemelor de clasificare.

Exemple de algoritmi:

- Support-vector machines
- Regresia liniară
- Regresia logistică
- Arbori de decizie
- Reţele neuronale
- Clasificator bayesian naiv

Pentru învătarea nesupervizată, procesul de antrenare constă în crearea unor modele interne de recunoaştere a unor tipare în urma analizei unui set de date neetichetat. Este deseori folosit în descoperirea similarităţilor şi diferenţelor într-un set de date.

Exemple de algoritmi:

- K-means clustering
- Autoencoders
- Analiza componentei principale
- Descompunerea valorilor singulare

În învăţarea prin recompensă, procesul de antrenare constă în maximizarea unei funcţii de recompensă, modelul calibrăndu-se astfel încat deciziile luate să ducă spre obţinerea unei recompense cât mai mari.

Exemple de algoritmi:

- Monte Carlo
- Q-learning
- SARSA

- Deep Q Network
- Proximal Policy Optimization
- Deep Deterministic Policy Gradient
- Trust Region Policy Optimization

## 1.3 Industrie

Tot mai multe aplicaţii folosesc tehnici de învăţare automată pentru optimizarea produselor, servicilor şi interacţiunilor cu utilizatorii. Cele mai notabile utilizări fiind:

- Algoritimi de căutare a ştirilor în baza unor preferinţe oferite explicit sau implicit de catre utilizator.
- Reclame personalizate generate după profilele utilizatorilor.
- Sisteme de recomandări produse.
- Etichetarea obiectelor sau persoanelor în imagini, înregistrări audio sau video.
- Sisteme robotice autonome.
- Maşini autonome.
- Sisteme meteorologice
- Sisteme de detectare a fraudelor într-un sistem bancar.
- Clasificare şi predicţia evenimentelor.
- Optimizarea proceselor de producţie a mărfurilor.
- Optimizarea procesului de antrenare pentru atleţi.

Companiile sunt foarte interesate de modul cum interacţionează şi percep clienţii produselor lor, ele încercând mereu să colecteze informaţii pentru despre modul cum sunt utilizate produsele în activitatea utilizatorului. Aceste campanii de colectare a datelor a devenit din ce în ce mai agresivă, marile companii software specializate în reţele sociale (Facebook, Twitter, Youtube, Linkedin, Reddit) vând datele utilizatorilor în vederea oferirii unui profil al consumatorului pentru a stabili interesul pentru produs.

## 1.4 Programe software pentru dezvoltare

Interesul puternic pentru acest domeniu a venit în principal din partea marilor companii software şi hardware, ele dezvoltănd puternice biblioteci pentru procesarea datelor, crearea de reţele neuronale, algoritmi de învăţare, etc. Pentru sprijinirea domeniului, aceste unelte sunt oferite dupa ca aplicaţii cu sursă deschisă ( în engleză *open source* ), avănd o licenţă deseori foarte permisibilă în vederea utilizări personale şi comerciale.

Calitatea acestor unelte le-a făcut să devină un standard în industrie, atăt comercială cât şi academică.

Example de biblioteci sau aplicaţii software:

- Tensorflow - bibliotecă dezvoltată de către Google în vederea utilizări cu usurinţă algoritmilor de învăţare, căt şi funcţii utilitare pentru manipularea datelor.
- PyTorch - bibliotecă dezvoltată de către Facebook pentru protiparea aplicaţilor de viziune computerizate, procesarea limbajului natural, etc.
- ML.NET - bibliotecă dezvoltată de Microsoft pentru crearea rapidă a unor aplicaţii de procesare a datelor folosind algoritmi de învăţare.
- scikit-Learn - bibliotecă care conţine funcţii statistice folosite pentru analiza datelor.
- Apache Spark - bibliotecă de aplicaţii destinate pentru procesarea unui volum foarte mare de date.
- Apache Kafka - aplicaţie care permite stocarea şi distribuirea unui volum foarte mare de date în timp real către mai mulţi consumatori.
- Caffe - bibliotecă pentru dezvoltare aplicaţilor pentru medii de lucru care nu dispun de o putere de procesare foarte mare, precum dispozitivele mobile.
- Keras - bibliotecă pentru dezvoltarea reţelelor neuronale
- H2O.ai - platformă de procesare şi analiză a datelor pentru mediul comercial

## 1.5 Big Data

O componentă esenţială pentru învăţarea automată este gestionarea datelor care vor fi folosite şi produse de către algoritmi algoritmii învăţare. Această gestionare a informaţiilor, de cele mai multe ori, va intra în cadrul domeniului de *Big Data*

Conform Uniunii Europene: ,,Big data se referă la volume de date colectate atât de mari şi complexe încât este nevoie de noi tehnologii, cum ar fi inteligenţă artificială, pentru a le procesa. Datele provin din nenumărate surse diferite.”[4]

Volumul de date pe care omenirea îl produce creşte de la an la an, cea ce face analiza şi intelegerea datelor să fie o sarcină din ce în ce mai dificilă. Tot mai mulţi oameni încep să

aibă acces la internet, iar numărul de dispozitive inteligente (smart phone, smart watch, smarth TV) pe care un individ de dispune creşte odată cu avansul tehnologic.

Principalele surse de provenienţă ale acestor date sunt:

- Reţele sociale - mesaje, imagini create de utilizatori pentru aşi exprima opinia la situaţia sociala, economică şi politică - datele pot fi utilizate pentru stabilirea unor tendinţe sociale cu privire la activitatea şi starea emoţională curentă şi viitoare a oamenilor.
- Mediul şi natura - date provenite de la sateliţi şi senzori pentru monitorizarea schimbărilor climatice - folosite pentru predicţia posibilelor dezastre naturale cauzate de activităţile omului.
- Sector public - documente, certificate, atestate, adeverinţe emise de către instituţile publice - pot fi utilizate în eficientizarea servicilor publice.
- Transport - date colectate prin GPS şi de la diferiţi operatori în domeniul transportului (transportul public, aeroporturi, gări) - pentru optimizarea rutelor şi a curselor de transport.
- Sector Medical - fişe medicale ale pacienţilor - monitorizarea stării de sănatatea a cetăţeinolor, utile pentru detectarea posibilelor amenţări de tip biologic.
- Iternetul Lucrurilor (*Internet of Things*) - date provenite de la diverse aparate, precum: telefon, ceas, televizor, senzor de gaz, sensor de umiditate, camere video, etc. - utilizate la monitorizare activităţii invidului cu scopul de a uşura anumite sarcini sau pentru a prevenii incidente.
- Sector industrial - reţele industriale de comunicaţii ( senzori, magistrale de teren, reţele celulare), rapoarte economice - folosite pentru automatizare şi îmbunătăţirea produselor şi a servicilor.
- Sector bancar - tranzacţii financiare, rapoarte - utilizate pentru detectarea fraudelor bancare, stabilirea ratelor la dobănzi, împrumuturi, schimb valutar, etc.

Toate aceste benificii sunt importante pentru societatea din ziua de astăzi, companii mare concureaza pentru crearea de infrastructură şi servicii pentru stocarea şi examinarea datelor.

Exemple de servicii:

- Amazon Web Services - cel mai mare furnizor de servicii şi infrastructură cloud din lume (având peste 200 de soluţii software).
- Microsoft Azure
- Google Cloud Platform

- IBM Cloud
- Oracle Cloud
- Alibaba Cloud

# Capitolul 2

# Reţele neuronale artificiale

## 2.1 Introducere

O reţea neuronală artificială este un model computaţional inspirat din structura şi modul de funţionare al creierului biologic. Conexiunile dintre neuronii artificiali se asemână sinapselor, fiecare neuron se conectează cu alt neuron prin intermediul unor muchii. Semnalul trimis prin aceste muchii este ponderat de nişte parametri numiţi ponderi sinaptice. Mai mulţi neuroni grupaţi formeaza un strat, iar mai multe straturi formează o reţea.

Procesul de învăţare presupune găsirea unor valori potrivite pentru ponderile sinaptice astfel încât procesarea semnalului de intrare să ofere rezultatul dorit.

## 2.2 Structură

Structura principală al unui neuron artificial este bazat pe modelul Perceptron-ului al lui Donald Hebb, modelul matematic fiind:

$$y = \varphi\left(\sum_{k=1}^{n} w_k * x_k + b\right)$$

,unde $x$ este vectorul de intrare (*input vector*), $y$ vectorul de ieşire (*output vector*), $w$ ponderea sinaptică (*weight*), $b$ deplasarea (*bias*) şi $\varphi$ este funcţia de activare sau transfer (*activation function*).

Vectorul de intrare este format din numerele reale, aceste numere putând reprezenta: imagini, frecvenţe, etichete codificate, valori provenite din senzori, etc. Ponderile sinaptice au rolul de a creşte sau descreste puterea semnalul reprezentat de valorile vectorului de intrare. Funcţia de activare preia semnalul ponderat şi oferă o valoarea spefică în baza acestuia. Deplasarea ajută la deplasarea semnalului ponderat pentru o mai bună aproximare necesară pentru îndeplirea anumitor condiţii ale funcţiei de activare.

**Exemplul 2.2.1** *Un neuron artificial care acţionează precum o poarta logică* **SAU***(OR)*

*pentru două numere binare are forma:*

$$y = \varphi(x_1 + x_2 - 0.5)$$

*,unde* $x = \{x_i | x_i \in \{0,1\}\}$, $y \in \{0,1\}$, $w_1 = 1$, $w_2 = 1$,$b = -0.5$, *iar funcţia de activare este:*

$$\varphi(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$$

**Verificare**. Pentru $x = [1,0]$, avem $u = 1 + 0 - 0.5 = 0.5$ şi $y = \varphi(u) = \varphi(0.5) = 1$ (acelaşi rezultat şi pentru $x = [0,1]$ - datorită propietaţii de comutativitate a adunării). Pentru $x = [1,1]$, avem $u = 1 + 1 - 0.5 = 1.5$ cu $\varphi(u) = \varphi(2) = 1$. Ultimul caz pentru $x = [0,0]$, vom avea $u =$

**Observaţia 2.1** *Fără funcţia de activare, perceptronul acţionează precum o funcţie liniară. Prin utilizarea unei funcţii de activare potrivite, puteam aborda mai uşor problemele neliniare, precum cele pentru clasificarea datelor în diverse categorii.*

Un singur perceptron oferă doar o singură valoare de ieşire. Dacă dorim să avem mai multe valori de ieşiri trebuie să mai adaugăm perceptroni. Gruparea de neuroni artificiali se numeşte *strat*.

Structura unui strat format din perceptroni arată astfel în formă matriceală:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \varphi(u_1) \\ \varphi(u_2) \\ \vdots \\ \varphi(u_n) \end{bmatrix}$$

Rezultatele acestui strat pot fi transmise către un alt strat care poate avea o altă funcţie de activare, astfel putem crea modele matematice mai complexe. Această înşiruire de straturi se numeşte *reţea*. Straturile intermediare sunt deseori referite ca *straturi ascunse*. Iar o reţea cu foarte multe straturi asunse poartă denumirea de *profundă (deep)*.

Reţele pot fi structurate şi sub forma unui graf. Fiecare neuron find repezentat de un nod, iar muchiile grafului sunt conexiunile dintre neuroni. Dacă graful suport nu conţine cicluri, spunem că este uni-directional - o denumire uzuală peste acest tip de reţea este *feed-forward (FF)* (denumire pe care o vom folosii şi în restul acestei lucrări). De asemenea, neuroni pot fi interconectaţi (graful suport conţine cicluri), fapt care poate oferii reţelei mai multă putere de modelare. Acest tip de reţea este denumit în general *recurrent neural network (RNN)*

Reţelele neuronale artificiale pot fi considerate ca find "aproximatori universali"[5]:

> Reţelele feed-forward multistrat sunt,în condiţii generale ale funcţiei de activare ascunsă, aproximatori universali dacă dispun de un număr suficient de unităţi asunse.

De-a lungul anilor, au fost create foarte multe tipuri de reţele neuronale artificiale pentru a servii la rezolvarea de probleme din domenii dificile.

Exemple de tipuri de reţele:

- Feed Forward (FF)
- Deep Feed Forward (DFF)
- Radial Basis Network (RBF)
- Recurrent Neural Network (RNN)
- Long/Short Term Memory (LSTM)
- Markov Chain (MC)
- Deep Convolutional Network (DCN)
- Deconvolutional Network (DN)
- Support Vector Machine (SVM)
- Deep Belief Network (DBN)

## 2.3 Funcţii de activare şi metode de optimizare

Funcţia de activare ajută reţeaua neuronală pentru înăţarea de tipare complexe aflate în setul de date analizat. Alegerea unei funcţii de activare este critică pentru performanţa reţelei, în special cazul problemelor neliniare.

Unele din cele mai folosite funcţii sunt:

$$
\begin{aligned}
\text{Identitate} \quad \varphi(x) &= x \\
\text{Binary Step} \quad \varphi(x) &= \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \\
\text{Logistic, Sigmoid} \quad \varphi(x) &= \frac{1}{1+e^{-x}} \\
\text{Rectified liniar unit(ReLU)} \quad \varphi(x) &= \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \\
\text{Softplus} \quad \varphi(x) &= \ln(1+e^{x})
\end{aligned}
$$

Funcţiile de activare pot aproape orice funcţie liniară sau neliniară, dar unele (precum cele enumerate anterior) oferă mai multe beneficii decăt altele în contextul antrenării unei reţele.

Crearea unei reţele neuronale artificiale presupune alegerea tipurilor de straturi din care să fie compusă împreună cu funcţile lor de activare. La început, ponderile sinaptice sunt de cele mai multe ori alese aleatoriu. Cea ce face ca aproximarea oferită de reţea să nu una foarte bună.

Acest lucru duce la problema de optimizare a reţelei (găsirea unor valori potrivite pentru ponderile sinaptice) astfel încât rezultatul aproximării să fie unul satisfăcător. Acest proces este referit uzual ca *antrenare (training)*. Antrenarea este unul dintre cele mai dificile capitole al acestui domeniu, alegerea unui algoritm potrivit poate beneficii extraordinarea în privinţa găsii unor valori optimale pentru ponderile sinaptice.

Reţelele neuronale artificiale folosite în industrie (aplicaţii de recunoaştere a obiectelor în imagini; programe pentru traduceri, recunoaşteri de voce ) au o complexitate extrordinară atăt din punct de vedere al arhitecturii, care poate constă dintr-un mix de diferite tipuri de reţele, cât şi al nivelului imens de date (de ordinul milioanelor de terabiţi). Antrenarea acestora poate dura câteva zile sau câteva luni, în cazuri rare fiind vorba de ani. Aşdar, un algoritm de optimizare eficient are un rol crucial în acest proces.

## 2.4 Tensorflow

Tensorflow este o platformă dedicată dezvoltării modelelor de învăţare automată. Acestă a fost creată iniţial de către Google pentru a accelara dezvoltarea domeniului prin oferirea de programe ajutătoare pentru crearea rapidă a protipurilor. Datorită calităţii superioare a programelor de prototipare şi a usurinţei de utilizare, acesta a devenit o platformă populară atât în mediul academic cât şi industrial.

Datorită popularităţii, platforma a beneficiat de multe contribuţii importante din partea marilor companii din domeniul IT şi cel al semiconductoarelor, precum: PayPal, AMD, nVIDIA, Blomberg, Intel, IBM, Qualcomm, Uber, Arm, Twitter. De asemenea, platforma beneficieaza de medii interactive de învăţare, ideale pentru studenţi sau profesionişti care doresc să dezvolte mici prototipuri de modele de învăţare automată.

Exemple de programe care fac parte din platforma Tensorflow:

- Tensorflow Hub - biblioteca care găzduieşte modele predefinite create de comunitate, precum: modele pentru clasificare imaginilor, analiza limbajului natural, generatoare de imagini
- Model Optimization - programe dedicate optimizării de modele
- TensorFlow Graphics - biblioteca care dispune de unelte pentru procesarea imaginilor
- TensorFlow Agents - biblioteca pentru dezolvatare agenţilor în cazul Învăţarii prin recompensă

În aceasta lucrare vom folosii aceste unelte pentru dezvoltarea unui model pentru agentul care va parcurge labirintul.

# Capitolul 3

# Metode de învăţare

## 3.1 Lanţ şi proces Markov

La baza modelului de învătare pe care il vom folosii în atrenarea agentului stau princiipile fundamentale ale lanţului Markov şi a procesului Markov. Propietatea Markov afirmă că viitorul depinde numai de prezent si nu de trecut. Un lanţ Markov este un model probabilistic care depinde numai de starea curenta pentru a prezice o stare viitoare. Aşadar, un lanţ Markov respectă propietatea Markov. Trecerea de la o stare la alta se numeşte tranziţie, iar probabilitatea ei poartă denumirea de probabilitate de tranziţie.

De cele mai multe ori, lanţul Markov este reprezentat sub formă de graf orientat al cărui muchii reprezintă probabilităţile de tranziţie dintre varfuri. Suma probailităţilor de tranziţie ale unui varf către celelalte varfuri este mereu 1.

In cazul agentului nostru, dacă ne-am imagina traseul ca find un lanţ Markov, atunci poziţiile din trase ar fi varfurile grafului, iar miscările agentului ar reprezenta muchiile. Putem asocia fiecarei mişcare o probabilitate, iar parcurgerea grafului reprezintă un posibil drum către obiectiv.

Aşadar, dacă ne-am dorii ca agentul să folosească aceasta idee pentru stabilirea unui drum pentru rezolvarea obiectivului trebuie să stabilim o strategie de parcurgere a grafului. O strategie simplă ar fi una de tip *Greddy*, şi anume agentul alege mereu acţiunea/muchia cu probabilitatea cea mai mare. Pentru ca aceasta strategie să aibă succes trebuie ca valorile probabilităţilor actîunilor să conducă către obiectiv în urma parcugerii grafului.

Peste lanţul Markov putem contrui un model matematic pentru modelarea procesului de decizie al agentului. Acest model conţine următoarele elemente:

- Mulţimea stărilor (S)
- Mulţimea acţiunilor (A)
- Probabilitatea de tranziţie dintr-o stare în alta pentru o acţiune ($P^a_{ss'}$)
- Probabilitatea de a primi o recompensă în urma tranziţiei ($R^a_{ss'}$)

- Factor de atenuare, pondere care exprimă importanţa recompenselor imediate şi viitoare ($\gamma$)

Pentru a controla comportamentul agentului în mediul de lucru, vom folosii un sistem de recompense pentru fiecare decizie luată. Dacă dorim ca agentul sa evite anumite situaţii precum luare de acţiuni care conduc la coliziuni cu anumite obiecte, vom asocia acestor decizii recompense negative. În contrast, dacă dorim ca agentul să facă anumite acţiuni care duc la îndeplinirea obiectivelor, aceste decizii vor fi asociate cu recompense pozitive.

Aşadar, dorim ca pe parcursul simulării agentul să acumuleze cât mai multe recompense pozitive şi să le evite pe cele negative. Acestu lucrul îl vom numi optimizare, iar procesul de antrenament implică modificarea reţelei neurale astfel încăt acţiunile rezultate din datele de intrare să maximizeze recompensele cumulate.

## 3.2 Ecuaţia Bellman

## 3.3 Q-Learning

# Capitolul 4

# Aplicaţie

## 4.1 Structură

Pentru contruirea aplicaţiei vom folosii următoare technologii:

- Svelte - program pentru construirea aplicaţilor web care va reprezenta interfaţa interactivă dedicată utilizatorului [6]
- Tensorflow.js - bibliotecă dedicată pentru contruirea şi antrenarea reţelelor neuronale pentru aplicaţii web [8]
- Echarts - bibliotecă pentru contruirea de graficelor dedicate vizualizarii de date [7]
- Konva - bibliotecă grafică pentru generarea imaginilor [9]

Labirintul este de forma unei matrici, fiecare celulă îndeplineşte un anumit rol: drum, obstacol, ieşire, etc. Clasa principala dedicată definirii structurii de reprezentare a labirintului este denumită **Board**.

Pentru codificare vom avea următoarele reguli: spaţiul liber va avea cifra 1, un obstacol cifra 2, iar pentru ieşire cifra 3. Pentru poziţile agentului, la codificarea celulei de matrice se va adauga prefixul 1. Exemple de definiri prin codificare se pot observa în tabelul 4.1.

| | | |
|---|---|---|
| 11 | 1 | 2 |
| 1 | 2 | 2 |
| 1 | 1 | 3 |

| | | |
|---|---|---|
| 1 | 1 | 2 |
| 11 | 2 | 2 |
| 1 | 1 | 3 |

| | | |
|---|---|---|
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 11 | 1 | 3 |

| | | |
|---|---|---|
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 1 | 11 | 3 |

| | | |
|---|---|---|
| 1 | 1 | 2 |
| 1 | 2 | 2 |
| 1 | 1 | 13 |

Tabelul 4.1: Exemple de codificări ale structurii labirintului

Aplicaţia este compusă din 3 parţi principale:

- Simulator - procesează comenzile agenţilor şi oferă date despre mediul simulat
- Interfaţa interactivă - permite vizualizarea mediul simulat şi interacţiunea cu agenţii

- Modele de învăţare - programe folosite pentru antrenarea agenţilor prin diferite tehnici folosind datele furnizate de către simulator

## 4.2 Simulator

Simulatorul actionează ca o interfaţa între agent şi mediul reprezentat de labirint prin clasa **Board**.

```
class Env {
    ACTIONS = ['UP', 'DOWN', 'RIGHT', 'LEFT']
    invalidState = false
    /**
     *
     * @param {Board} board
     */
    constructor(board) {
        this.board = board
    }

    //
    setAgentStartPosition(pos) {
        this.board.playerDefaultPos = pos
    }

    //
    step(action) {
        this.invalidState = !this.board.move(this.ACTIONS[action])
        return  this.board.getBoardState(), this._getReward(), this._isDone
    ()]
    }

    //
    reset() {
        this.board.playerReset()
        return this.board.getBoardState()
    }

    //
    actionSample() {
        return Math.floor(Math.random() * this.ACTIONS.length)
    }

    //
    _getReward() {
        return (this.invalidState && !this.board.isOnExit() && -100) ||
    this.board.getPlayerCellValue()
    }

    //
    _isDone() {
```

```
        return this.board.isOnExit() || this.invalidState
    }

    //
    clone() {
        return new Env(this.board.clone())
    }
}
```

## 4.3 Interfaţă

## 4.4 Agent

```
class Agent {
    constructor(model) {
        /**
         * @type {tf.Sequential}
         */
        this.model = model || this.buildModel()
    }

    buildModel() {
        const model = tf.sequential()

        model.add(tf.layers.dense({ units: 10, inputShape: [10, 10], activation: 'relu' }))
        model.add(tf.layers.flatten())
        //model.add(tf.layers.dense({ units: 8, activation: 'relu' }))
        //model.add(tf.layers.dense({ units: 16, activation: 'relu' }))
        //model.add(tf.layers.dropout({ rate: 0.2 }))
        // model.add(tf.layers.dense({ units: 32, activation: 'relu' }))
        model.add(tf.layers.dropout({ rate: 0.2 }))
        model.add(tf.layers.dense({ units: 4, activation: 'linear' }))
        model.compile({ loss: 'meanSquaredError', optimizer: 'adam', metrics: ['accuracy'] })
        model.summary()
        return model
    }

    async fit(input, output) {
        await this.model.fit(tf.tensor3d([input]), output, { epochs: 5 })
    }

    predict(input) {
        return this.model.predict(tf.tensor3d([input]))
    }

    getAction(input) {
        const result = this.predict(input)
```

```
35          return tf.argMax(result, 1).arraySync()[0]
36      }
37  }
```

## 4.5 Model de învăţare

# Concluzii finale

În această lucrare am analizat ....

# Bibliografie


[1] Hebb, D. O. The organization of behavior : a neuropsychological theory / D.O. Hebb, Wiley New York, 1949

[2] http://infolab.stanford.edu/pub/voy/museum/samuel.html

[3] https://www.ibm.com/cloud/learn/unsupervised-learning

[4] https://www.europarl.europa.eu/news/ro/headlines/society/20210211STO97614/big-data-def

[5] Kurt Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 1991

[6] https://svelte.dev

[7] http://echarts.apache.org/en/index.html

[8] https://www.tensorflow.org/js

[9] https://konvajs.org/docs/index.html

[10] Cormen, T.H., Leiserson, C.E., Rivest, R.L. *Introducere în algoritmi*, Cluj-Napoca, Editura Computer Libris Agora, 2000.

[11] https://www.mathworks.com/products/computer-vision.html

# Index

capitol
    C1, 7
    C2, 9
concluzii, 11

sectiune
    S1.1, 7