Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Automatică și
Informatică Aplicată

CLASIFICAREA SEMNALELOR EEG FOLOSIND REȚELE CONVOLUȚIONALE PENTRU DETERMINAREA STĂRII MENTALE

Proiect de Diplomă

Năsui Alexandru-Andrei

Conducător științific
S.l.dr.ing. **Ana Maria DAN**

Timișoara
Iunie, 2020

Cuprins

Capitolul 1

Introducere

Ceva despre istoria AI

1.1 Temă. Obiective. Motivație

Tehnicile de învățare automată urmăresc crearea unor modele matematice bazate pe seturi de date inițiale, denumite *seturi de antrenare (training data)*, care pot generaliza informațiile din acestea, iar mai apoi să prezică răspunsul pentru seturi de date necunoscute. Învățarea automată este folosită intr-o largă gamă de aplicații, precum filtrarea mesajelor e-mail de tip spam de cele autentice, răspunsurile date de către motoarele de căutare, clasificarea celulelor tumorale in benigne sau maligne, recunoașterea facială, recunoașterea diverselor obiecte, recunoașterea limbajului vorbit și scris, și mai nou la conducerea automată a mașinilor.

Cele mai multe tehnici de învățare automată fac parte din una dintre cele trei categori:

- Învățare supervizată (Supervised Learning)
- Învățare nesupervizată (Unsupervised Learning)
- Învățare cu întărire (Reinforcement Learning)

Învățarea supervizată

Învățarea supervizată, în momentul de fața este cea mai răspandită metodă folosită în practică. Principiul din spatele acesteia constând în construirea unui model matematic, prin diferite tehnici, bazat pe un

set de date etichetate. Acest set de date etichetate este alcătuit din înregistrări care reprezintă o corespondență intre atribute (intrări) si o clasă (ieșire). Astfel, se urmărește generalizarea acestor corespondețe si posibilitatea prezicerii clasei unei înregistrări care nu aparține de datele folosite la învățare. Unii dintre cei mai folosiți algoritmi de învățare supervizată sunt:

- Arbori de decizie
- Metode de regresie
- Algoritmi genetici
- Rețele neuronale artificiale
- Mașini cu vector suport
- Rețele Bayesiene

Învățarea nesupervizată

Procesul de învățare nesupervizată diferă față de cel amintit anterior prin faptul că acesta folosește un set de date de antrenare neetichetat. Algoritmii primesc doar un set de atribute (date de intrare), ne știind ieșirea asociată acestora. Aceștia caută in aceste date asemănări și deosebiri, bazându-se pe proprietățile statistice a datelor. Printre cele mai răspândite tehnici se numără:

- Tehnici de grupare
  - Grupare ierarhizată
  - Tehnica k-means
- Hărți cu auto-organizare
  - Rețele Kohonen
- Modele Markov cu stări invizibile

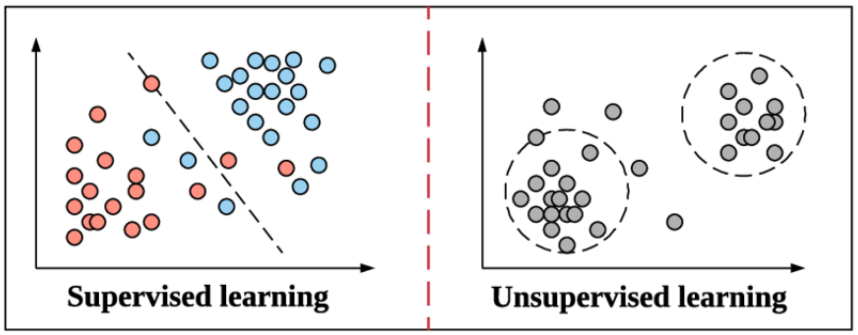


Figura 1.1: Diferența dintre modul de funcționare al învățării supervizare și învățării nesupervizate [1]

Învățarea cu întărire

Învățarea cu întărire este o metodă de învățare prin interacțiuni repetate a unui *agent (software agent)* cu mediul, cu urmărirea atingerii unui anumit scop (Figura 1.2). Interacțiunile se bazează pe acțiunile luate de agent la stimulii mediului, pentru care v-a primit o recompensă de la acesta în funcție de beneficiul adus îndeplinirii scopului. Recompensele primite au rolul de a îmbunătății capacitatea agentului de a lua cea mai bună decizie din starea în care acesta se afla la momentul acțiunii. Scopul pe termen lung al agentului este maximizarea numărului de recompense primite. După repetate interacțiuni cu mediul, capacitatea agentului de a lua decizii bune se v-a crește, iar drumul acestuia prin mediu va tinde spre optim.

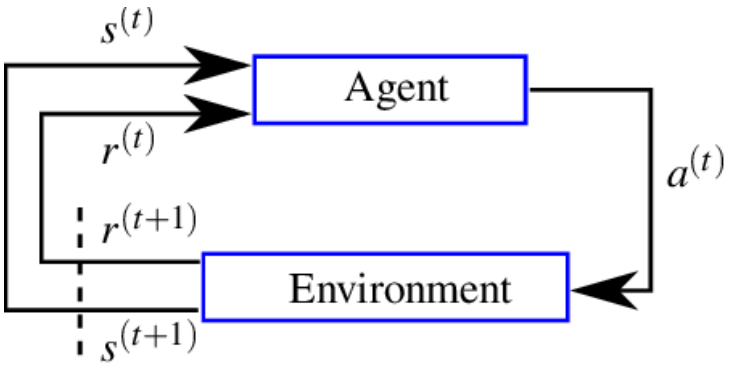


Figura 1.2: Modul de funcționare al învățării cu întărire [2]

Obiective

Această lucrare își propune folosirea rețelelor convoluționale pentru clasificarea a trei clase/stării mentale diferite. Metoda de clasificare se bazează pe reprezentarea informațiilor statistice și spectrale a undelor cerebrale sub forma unor imagini alb-negru. Datele EEG *(Electroencefalograma)* au fost extrase cu ajutorul căștii valabile comercial, Muse 2016. Datele extrase au fost prelucrate și etichetate, rezultând 414 atribute și clasa de care aparțin, neutru, relaxat sau concentrat. După extragerea atributelor, au fost selectate și normalizate în intervalul [0:1] 400 de atribute pentru a putea reprezenta o imagine alb-negru de dimensiunea 20x20. După antrenarea clasificatorului cu aceste imagini, acuratețea acestuia la prezicerea imaginilor care nu se aflau in setul de date de antrenare, a fost de 90%.

1.2 Soluții existente

Apariția unor soluții comerciale *low-cost* non-invazive a facut posibilă înregistrarea și analiza undele cerebrale în afara domeniului medical [3]. În principal, aceste dispozitive sunt folosite în activități simple de interfațare a creierului cu calculatorul *(BCI - Brain-Computer Interface)*. Raportul zgomot-semnal util mare și eșantionarea imperfectă reprezentând dezavantajele acestor aparate comparativ cu cele de nivel medical. Acest lucru însă nu a împiedicat apariția a tot mai multor studii care folosesc aceste aparate ca o resursă [4].

EEG[5]
EEG-CNN[6]
TODO_PLACEHOLDER

- Utilizare casca pt clasificare EEG de detectare a starii

  Scriu despre articolul de clasificare EEG de la cei care au scris articolul cu CNN

  O sa scriu despre articolul cu CNN

1.3 Structurare pe capitole

Lucrarea este structurată după cum urmează. *Capitolul 2* prezintă o introducere a ceea ce înseamna și modul în care funcționează rețelele

neuronale de tipul *feedforward* ca mai apoi să poată fii folosite ca suport în înțelegerea funcționalității rețelelor convoluționale. V-or fii menționate avantajele și prezentate aplicațiile de succes ale acestora. Tot aici, v-or fii prezentate bazele undelor cerebrale, ce sunt acestea, cum funcționează și felul în care pot fii detectate. În *Capitolul 3* este prezentat modul de implementare al soluției împreună cu detaliile tehnice aferente. Explicarea tehnicilor de extragere și prelucrare a datelor, detalii privind arhitectura rețelei folosite și rezultatele produse de aceasta se v-or regăsi la finalul acestui capitol. Finalul lucrării conține *Capitolul 4*, care aduce concluzii legate de tema acestei lucrări.

Capitolul 2

Studiul teoretic. Tehnologii folosite

TODO O scurta introducere cu privire la ce reprezinta acest capitol
TODO Scrie in introducere si despre undele cerebrale!
TODO Rescrie textul sa fie mai frumos spusa toata informatia + diacritice

Acest capitol are rolul de a prezenta aspectele fundamentale ale unei retele neuronale si construirea retelelor convolutionale din acestea. Pornind de la perceptron, element de baza, folosirea mai multor neuroni pentru a creea o retea neuronala iar mai apoi, trecerea de la retelele neuronale simple, la retelele convolutionale.

2.1 Rețele neuronale

Rețelele neuronale artificiale reprezintă un sistem de calcul inspirat după modelul rețelelor neuronale biologice, atât ca și structură, cât și ca mod de procesare a informației. Neuronul artificial reprezintă unitatea elementara a unui rețele neuronale artificiale. Precum neuronul biologic, care conține dendrite și axioni, neuronul artificial imită această structură prin noduri de intrare si noduri de ieșire.

2.1.1 Perceptronul

Conceptele fundamentale ale neuronului artificial provin din modelul perceptronului introdus de către *Frank Rosenblatt (1958)* [7] pe baza cercetarilor anterioare ale lui *Warren McCulloch* și *Walter Pitts* [8].

Perceptronul primește o serie de valori la intrare, $x_1, x_2, x_3 \ldots$, și produce o singură ieșire. Pentru calculul ieșirii, sunt folosite așa numitele *ponderi (weights)*, notate cu $w_1, w_2, w_3, \ldots$, numere reale reprezentând importanța fiecărei intrări în determinarea ieșirii. Astfel,

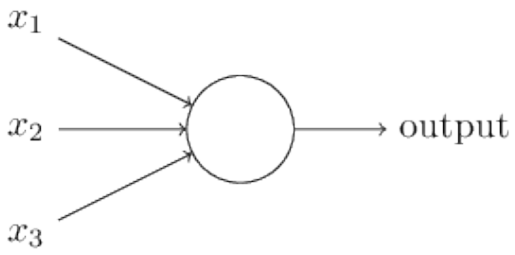


Figura 2.1: Reprezentarea grafică a unui perceptron [9]

ieșirea perceptronului, este determinată de suma ponderată $\sum_i x_i w_i$ comparată cu o valoare reală numită *prag*. Această comparație reprezintă o funcție pentru determinarea ieșirii y, denumită *funcție de activare*. Matematic, funcția de activare a perceptronului este reprezentată de o formă discretă a funcției *treaptă unitate* (Fig. 2.3b).

$$y = \begin{cases} 0 & : \sum_i x_i w_i \leq prag \\ 1 & : \sum_i x_i w_i > prag \end{cases} \tag{2.1}$$

Putem simplifica modul prin care perceptronii sunt descriși, rescriind $\sum_i x_i w_i$ ca fiind produsul cartezian $x \cdot w$, unde x și w sunt vectori care conțin intrările x_i respectiv ponderile w_i. A doua modificare pe care o putem face este să mutăm termenul $prag$ în partea stângă a inegalității, denumindu-l *bias/offset*, $b = -prag$. Astfel, ecuația 2.1 devine:

$$y = \begin{cases} 0 & : x \cdot w + b \leq 0 \\ 1 & : x \cdot w + b > 0 \end{cases} \tag{2.2}$$

Bias-ul poate fii considerat ca o intrare suplimentară de valoate $x_0 = 1$ și $w_0 = b$ care permite translatarea funcției de activare la stânga sau la dreapta.

2.1.2 Perceptronul multistrat

Prin conectarea mai multor perceptroni rezultă rețeaua numită „perceptronul multistrat” *(Multi-layer Perceptron - MLP)*. Aceasta este formată în general de o succesiune de straturi de perceptroni complet conectați, compusă dintr-un strat de intrare, unul sau mai multe straturi ascunse *(hidden layer)* și un strat de ieșire. Ieșirile unui strat reprezintă intrările pentru stratul următor. Rețeaua care conține mai multe straturi ascunse poartă denumirea de *„rețea neuronală adâncă” ("deep neural network")*, iar în cazul in care aceasta conține un singur strat ascuns poartă denumirea de *„rețea neuronală superficială” ("shallow neural network")*. Figura 2.2 prezintă o astfel de rețea.

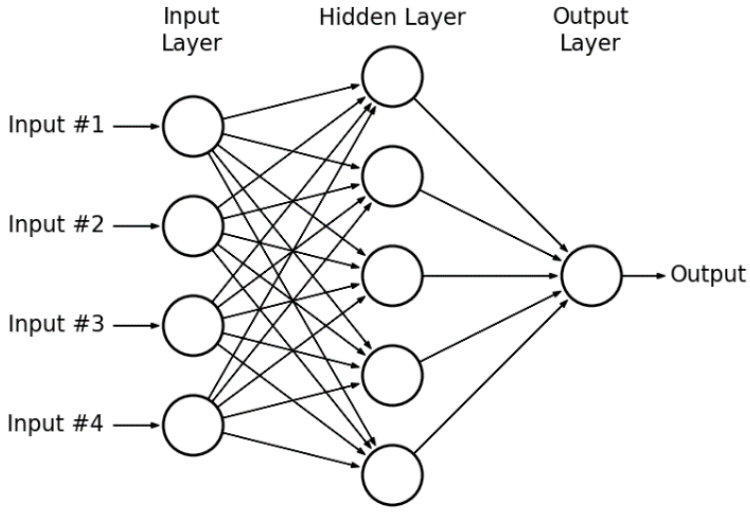


Figura 2.2: Rețea de tipul perceptron multistrat [10]

Rețeaua prezentată mai sus este de tip *propagare înainte (feed-forward)*, adică, informația în rețea circulă intr-o singură direcție, de la stânga la dreapta. Rezultatul procesării informațiilor de către primul strat de neuroni va reprezenta intrarea pentru stratul al doilea, astfel ieșirile acestuia având o semnificație mai abstractă și complexă comparativ cu primul strat. Cu fiecare strat ascuns adăugat rețelei, nivelul de abstractizare al informației va crește, astfel deciziile luate de rețea devenind tot mai sofisticate.

Limitarea acestui tip de rețea poate fii observată încercând să aplicăm schimbări mici ponderilor w conexiunilor (sau a $bias$-ului) unui strat pentru a obține o schimbare mică a ieșirii rețelei. Analitic,

acest lucru se rezumă la următoarea ecuație:

$$\Delta y \approx \sum_i \frac{\partial y}{\partial w_i} \Delta w_i + \frac{\partial y}{\partial b} \Delta b \tag{2.3}$$

În realitate însă acest lucru nu se întâmplă întotdeauna. Aceste mici modificări pot determina schimbarea complet a stării[1], spre exemplu de la 1 la 0. Acest comportament poate declanșa o schimbare foarte complicată și greu de controlat în întreaga rețea.

Limitarea dată de capacitatea perceptronului de clasificare binară, poate fii rezolvată însă folosind un alt tip de funcție de activare.

2.1.3 Neuronul

Neuronul artificial sigmoid, este foarte asemănător cu perceptronul prezentat anterior. Acesta este format dintr-un vector de intrări x, un vector al ponderilor w, un $bias$ și o ieșire. Valorile vectorului de intrare x nu sunt însă limitate la valorile 1 sau 0, neuronul sigmoid fiind capabil să proceseze valori reale. Precum x, ieșirea y a neuronului poate lua valori reale, fiind dată de ecuația *funcției sigmoid* prezentată mai jos:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \text{, unde } z = w \cdot x + b \tag{2.4}$$

Analizând graficele din Figura 2.3 putem observa faptul că funcția sigmoid este defapt o versiune netezită a funcției treaptă unitate. Acest lucru ne asigură că schimbările mici efectuate atât in vectorul ponderilor w cât și în $bias$ v-or fii reflectate în ieșire și că nu v-om avea salturi bruște de la 0 la 1 la ieșirea neuronului. Totuși, modelul perceptronului poate fii simulat folosind funcția sigmoid. Atunci când $z \to \infty$, $\sigma(z) \approx 1$, iar când $z \to -\infty$, $\sigma(z) \approx 0$. Folosind astfel de funcții de activare ne ajută să găsim ponderile potrivite mult mai ușor și putem afla felul în care modificările acestora afectează ieșirea.

În general, se folosesc funcții derivabile pe întreg domeniul de definiție, care nu au treceri bruște de la un capăt la altul, pentru a facilita antrenarea rețelei. În practică se mai folosesc funcții precum *tangenta hiperbolică*:

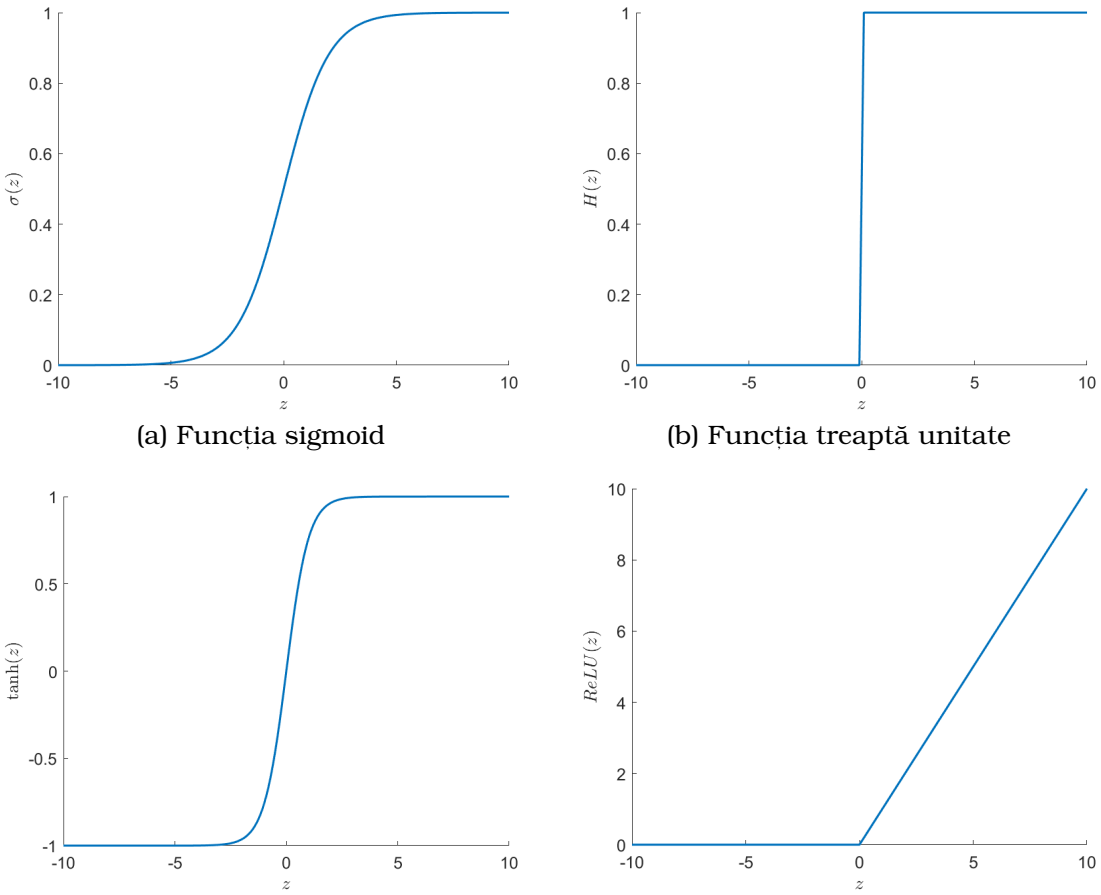
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.5}$$

[1]Acest lucru poate fi observat în graficul funcției de activare al perceptronului din Figura 2.3

sau *funcția unitate liniară rectificată* (**ReLU**):

$$f(z) = \max(0, z) \tag{2.6}$$

Mai jos sunt prezentate graficele diferitor funcții de activare.



(a) Funcția sigmoid

(b) Funcția treaptă unitate

(c) Funcția tangenta hiperbolică

(d) Funcția unitate liniară rectificată

Figura 2.3: | **TODO** Fontu la label mai mare, din MATLAB | Funcții de activare

Este necesar de menționat faptul că aceste funcții se folosesc în special pentru neuronii aflați în straturile ascunse ale rețelei, stratul de ieșire folosind de obicei funcții logistice pentru clasificări binare și funcția *softmax* pentru clasificări multi-clasă.

2.1.4 Antrenarea rețelelor

Algoritmul prin care se realizează antrenarea rețelelor de tipul feedforward, poartă denumirea de *algoritm de propagare înapoi (back-propagation)*. Acest algoritm a fost făcut faimos de către David Rumelhart, Geoffrey Hinton, și Ronald Williams în 1986 [11]. Algoritmul constă în modificarea repetată a ponderilor conexiunilor din rețea în încercarea de a minimiza o funcție care reprezintă eroarea dintre rezultatul așteptat și cel obținut. Funcțiile folosite cu scopul de a fii minimizate se numesc *funcții obiectiv* sau *funcții de cost* (§2.1.5).

Ecuațiile algoritmului back-propagation

Algoritmul clasic de back-propagation a fost inițial conceput să rezolve probleme de regresie folosind funcții de activare sigmoide. Totuși, acesta poate fii aplicat și problemelor de clasificare cu sau fară astfel de funcții de activare. Pentru a putea face o descriere matematică asupra modului de funcționare al algoritmului, este necesară folosirea următoarelor notații [9]:

1. C pentru a reprezenta o funcție de cost
2. w^l_{jk} pentru specificarea ponderii conexiunii de la neuronul k în stratul $l-1$ către neuronul j în stratul l.
3. b^l_j pentru bias-ul neuronului j din stratul l
4. a^l_j pentru activarea neuronului j din stratul l

a^l_j depinde de activarea neuronului din stratul $l-1$ conform ecuației

$$a^l_j = \sigma(\sum_k w^l_{jk} a^{l-1}_k + b^l_j) \tag{2.7}$$

unde k reprezintă toți neuronii din stratul $l-1$. Folosind aceste notații pot fii descrise cele patru ecuații fundamentale ale acestui algoritm:

1. Ecuația pentru determinarea erorii din stratul de ieșire L

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j) \tag{2.8}$$

2. Ecuație pentru determinarea erorii din stratul l în raport cu eroarea din stratul $l+1$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{2.9}$$

3. Ecuație pentru determinarea ratei de schimbare a funcției de cost în funcție de orice bias din rețea

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{2.10}$$

4. Ecuație pentru determinarea ratei de schimbare a funcției de cost în funcție de orice pondere din rețea

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{2.11}$$

Aplicarea algoritmului back-propagation

Inițial toți parametrii rețelei (ponderile w și bias-ul b) v-or fii inițializați aleator[2]. Urmează apoi etapa de propagarea înainte a datelor din setul de date de antrenare și a comparării rezultatului prezis în comparație cu cel real. Această eroare se calculează conform ecuației (2.8), iar mai apoi va fii propagată înapoi în rețea, strat cu strat, începând cu stratul $l = L - 1$, conform ecuației (2.9). Parametrii v-or fi ajustați în funcție de gradientul funcției de cost conform ecuațiilor (2.10) și (2.11). Algoritmul se repetă pentru un numar stabilit de pași (aleși experimentali) sau până când eroarea δ^L scade sub o valoare impusă. Odată cu finalizarea algoritmului, rețeaua va fii pregătită să proceseze date pe care nu le-a întâlnit în setul folosit la antrenare și să facă predicții asupra acestora.

2.1.5 Funcții de cost

O funcție de cost este o funcție de forma [9]

$$C(w, b, S^T, E^T) \tag{2.12}$$

[2]Nu este neapărat ca parametrii rețelei să fie inițializați aleator, existând mai multe metode de inițializare

unde w și b reprezintă parametrii rețelei S^T intrarea unui eșantion de date de antrenare și E^T ieșirea dorită din eșantionul respectiv.

Rezultatul dat de asemenea funcții este un număr care reprezintă performanța rețelei de a face preziceri pe baza modificărilor aduse parametrilor acesteia. Se urmărește prin diferite *tehnici de optimizare* minimizarea acestei funcții, valoarea returnată purtând denumirea de *eroare/cost/loss*. În anumite situații însă, spre exemplu în cazul învățării cu întărire, scopul este de a maximiza această funcție, rezultatul fiind denumit *recompensă*.

Atât funcțiile de cost, cât și tehnicile de optimizare ale acestora trebuie alese în funcție de problema în cauză, neexistând o soluție universal valabilă. Empiric, funcțiile de cost se pot împărți în două categorii:

- **pentru probleme de regresie**: eroarea medie absolută *(L1 loss)*, eroarea medie pătratică *(L2 loss)*, Huber Loss
- **pentru problemele de clasificare**: funcția de cost logaritmică *(cross-entropy)*, categorical cross-entropy, divergența Kullback–Leibler

2.2 Rețele convoluționale

2.3 Undele cerebrale

TODO Edit title

Capitolul 3

Prezentarea aplicației

3.1 Etape implementare

3.1.1 Preluare date

3.1.2 Prelucrare date

3.1.3 Algoritm

3.2 Rezultate

Capitolul 4

Concluzii

Bibliografie

[1] Bin Qian, Jie Su, Zhenyu Wen, Renyu Yang, Albert Zomaya, and Omer Rana. Orchestrating development lifecycle of machine learning based iot applications: A survey. 10 2019.

[2] Christopher Gatti and M. Embrechts. Reinforcement learning with neural networks: Tricks of the trade. *Studies in Computational Intelligence*, 410:275–310, 01 2013.

[3] EMOTIV. What is an eeg headset? definition & faqs. `https://www.emotiv.com/glossary/eeg-headset/`.

[4] Alejandro Morán and Miguel C. Soriano. Improving the quality of a collective signal in a consumer eeg headset. *PLOS ONE*, 13(5):1–21, 05 2018.

[5] Jordan Bird, Luis Manso, Eduardo Ribeiro, Aniko Ekart, and Diego Faria. A study on mental state classification using eeg-based brain-machine interface. 09 2018.

[6] Jodie Ashford, Jordan Bird, Felipe Campelo, and Diego Faria. *Classification of EEG Signals Based on Image Representation of Statistical Features*, pages 449–460. 01 2020.

[7] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[8] W S McCulloch and W Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943.

[9] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[10] Hassan Hassan, Abdelazim Negm, Mohamed Zahran, and Oliver Saavedra. Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake. *International Water Technology Journal*, 5, 12 2015.

[11] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.