Notițe curs

-Inteligență artificială-

Curs 1: Sisteme inteligente

Tipologie:

- În funcție de **experiența acumulată** în timpul învățării:
  - cu învățare supervizată
  - cu învățare nesupervizată
  - cu învățare activă
  - cu învățare cu întărire
- În funcție de **modelul învățat** (algoritmul de învățare):
  - Rețele neuronale artificiale
  - Mașini cu suport vectorial (MSV)
  - Algoritmi evolutivi
  - kNN
  - Arbori de decizie
  - Modele Markov ascunse

Predicții / regresii:

- Scop: predicția output-ului pentru un input nou folosind un model învățat anterior
- Ex.: predicția vânzărilor dintr-un produs pentru un moment de timp viitor în funcție de preț, lună calendaristică, regiune, venit mediu pe economie.

Regresii simbolice:

- Scop: estimarea formei unei funcții uni sau multivariate folosind un model învățat anterior
- Ex.: estimarea funcției care modelează conturul unei suprafețe

Clasificare:

- Scop: clasificarea formei unui obiect într-una sau mai multe categorii (clase) - cunoscute anterior sau nu - pe baza caracteristicilor (atributelor, proprietăților) lui
- Ex.: sistem de diagnoză pentru un pacient cu tumoare: nevasculară, vasculară, angiogenă

Planificare:

- Scop: generarea unei succesiuni optime de acțiuni pentru efectuarea unei sarcini
- Ex.: planificarea deplasării unui robot de la o poziție dată până la o sursă de energie (pentru alimentare)

Învățare automată:

1. Învățare supervizată:
    a. Ex.: regresie, clasificare
    b. Caracteristici:
        i. Date etichetate (se cunosc o parte din datele de intrare și ieșire)
        ii. Feedback direct în timpul învățării (algoritmul se adaptează la datele de intrare și ieșire)
        iii. Predicție a datelor de ieșire (fiind cunoscute niște date de intrare diferite de cele inițiale)
2. Învățare nesupervizată:
    a. Ex.: clusterizare, reducerea numărului de dimensiuni
    b. Caracteristici:
        i. Date neetichetate (se cunosc o parte din datele de intrare)
        ii. Fără feedback direct în timpul învățării (pentru că nu se cunosc datele de ieșire)

iii. Identificarea unor structuri în date (generarea de date de ieșire pentru datele de intrare inițiale)

3. Învățare prin întărire:
   a. Caracteristici:
      i. Predicția unor secvențe de decizii/acțiuni
      ii. Sistem de recompense (pentru fiecare decizie)
      iii. Se învață un model de acțiune (o serie de acțiuni ce trebuie efectuate)

Învăţare supervizată

- Calitatea învăţării → Măsuri de performanţă → Măsuri statistice
  - Eroarea de predicție
    - Suma diferențelor absolute între valorile reale și cele calculate

$$Err = \frac{1}{noSamples} \sum_{i=1}^{noSamples} abs(real_i - computed_i)$$

    - Suma pătratelor diferențelor între valorile reale și cele calculate

$$Err = \sqrt{\frac{1}{noSamples} \sum_{i=1}^{noSamples} (real_i - computed_i)^2}$$

TP - true positive

FN - false negative

Acuratețea = TP / (TP + TN + FP + FN)

Precizia (P) = TP / (TP + FP)

Rapelul (R) = TP / (TP + FN)

Scorul F1 = 2 * PR / (P + R)

Curs 2: Sisteme care învață singure

Metoda celor mai mici pătrate

- Presupunem cazul unei probleme de regresie
  - Date de intrare $x^i \in R^d$, i=1,n
  - Date de ieșire $y^i \in R$
  - Se cere un model **liniar** f care transformă orice x^i în y^i , i=1,n
  - $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d$

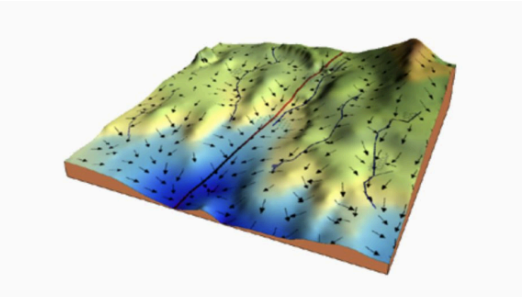
  - Se poate defini o funcție de cost
  - Loss = $\sum_{i=1,n} (y^i - f(x^i))^2$ -- minimizată ➔ valorile optime ale lui β

  - Derivarea loss-ului după β : $\beta = (X^T X)^{-1} X^T y$

  - Daca d = 1, $\beta_1 = cov(x,y)/var(x)$, $\beta_0 = y - \beta_1 x$

Metoda gradient descent

- Presupunem cazul unei probleme de regresie
  - Date de intrare $x \in R^d$
  - Date de ieșire $y \in R$



  - Se cere un model **liniar** f care transformă x în y
  - $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d$
  - Invățare supervizată

- Modelarea coeficienților β:
  - la iterația 0: valori random (sau 0)
  - la iterația t + 1 (t = 0, 1, 2, …)

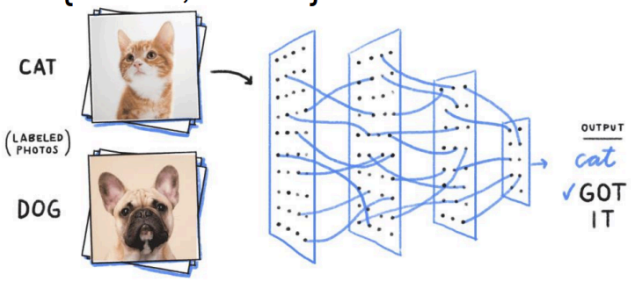
  $\beta_k(t+1) = \beta_k(t) - learning\_rate * error(t) * x_k$, k=1,2,...,d

  $\beta_0(t+1) = \beta_0(t) - learning\_rate * error(t)$
    - Unde
      - error(t) = computed - realOutput
      - $error(t) = \beta_0(t) + \beta_1(t)*x_1 + \beta_2(t)*x_2 + ... + \beta_d(t)*x_d - y$

- Stochastic Gradient Descent
  - - eroarea se calculează pentru fiecare exemplu de antrenament
  - - modelul se updatează pentru fiecare exemplu de antrenament
- Batch Gradient Descent
  - - eroarea se calculează pentru fiecare exemplu de antrenament
  - - modelul se updatează după ce toate exemplele de antrenament au fost evaluate (la finalul unei epoci)
- Mini-Batch Gradient Descent (combinare a celor două)
  - - setul de date se împarte în mai multe părți
  - - eroarea se calculează pentru fiecare exemplu de antrenament dintr-un mini-batch
  - - modelul se updatează pentru fiecare exemplu de antrenament dintr-un mini-batch

Regresie logistică

- Presupunem cazul unei probleme de clasificare
  - Date de intrare $x^i \in R^d$, i=1,n
  - Date de ieșire $y^i \in \{0,1\}$ sau {label1, label2}



  - Se cere un model **liniar** f care separa orice x^i în 2 clase (etichetate cu 0 și 1)
  - $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d$
  - Invățare supervizată

- Regresie Logistică (clasificare)
  - Mapează datele intr-un set discret de clase (label-uri)
  - Tipuri:
    - Binar (Pass/Fail, True/False)
    - Multi (Cat,Dog,Panda)
    - Ordinal (Low,Medium,High)
  - Folosește funcția sigmoid pentru a decide clasa de apartenență
  - Putem folosi gradient descent pentru minimzarea erorii

Funcția sigmoid: $S(z) = \frac{1}{1 + e^{-z}}$ (mapează orice număr real în (0, 1))

- Modelarea coeficienților β:
  - la iterația 0: valori random (sau 0)
  - la iterația t + 1 (t = 0, 1, 2, …)

  $\beta_k(t+1) = \beta_k(t)$ - learning_rate * error(t) * x_k, k=1,2,...,d

  $\beta_0(t+1) = \beta_0(t)$ - learning_rate * error(t)
    - Unde
      - error(t) = Sigmoid(computed) - realOutput
      - error(t) = Sigmoid($\beta_0(t) + \beta_1(t)*x_1 + \beta_2(t)*x_2 + ... + \beta_d(t)*x_d) - y$

- Clasificarea rezultatelor
  - (0,1) -> [$label_0$, $label_1$, .. $label_n$]

Regresia liniară cu metodele sunt detaliate în PDF-ul "Regresia Liniară"!!!

Curs 3: Mașini cu suport vectorial (MSV)

Definire:

Mașinile cu Suport Vectorial sunt algoritmi de învățare automată supervizată folosiți pentru clasificare și regresie.

Ele construiesc un hiperplan sau un set de hiperplane într-un spațiu cu dimensiuni mari sau infinite, care sunt folosite pentru separarea diferitelor clase.

- MSV găseşte o funcţie liniară de forma $f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$, (**w** -vector pondere) a.î.

$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

- $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ → hiperplanul de decizie care separă cele 2 clase

Tipuri de probleme rezolvabile:

- Clasificare binară
  - MSV sunt adesea folosite pentru clasificarea binară, unde scopul este de a separa datele în două clase distincte.
- Clasificare Multiclasă:
  - MSV pot fi extinse pentru clasificarea multiclasă folosind tehnici cum ar fi "one-vs-one" sau "one-vs-rest"

One-Vs-One (OvO):

Se antrenează un clasificator binar pentru fiecare pereche de clase. Dacă există *k* clase, se vor antrena $\frac{k(k-1)}{2}$ clasificatori.

Construire clasificatori:

- Pentru fiecare pereche de clase, se construiește un clasificator binar care decide între cele două clase.
- De exemplu, pentru clasele A, B și C, se vor construi clasificatori pentru perechile (A, B), (A, C) și (B, C).

Clasificare:

- Pentru un nou exemplu, fiecare clasificator binar emite o predicție
- Exemplul este clasificat în clasa care primește cele mai multe voturi dintre toți clasificatorii binari.

One-Vs-Rest (OvR):

Se antrenează un clasificator binar pentru fiecare clasă, unde fiecare clasificator învață să distingă o clasă de toate celelalte clase combinate.

Construire clasificatori:

- Pentru fiecare clasă, se construiește un clasificator binar care învață să recunoască acea clasă în contrast cu toate celelalte clase.
- De exemplu, pentru clasele A, B și C, se vor construi clasificatori pentru A vs (B, C), B vs (A, C) și C vs (A, B)

Clasificare:

- Pentru un nou exemplu, fiecare clasificator binar emite o predicție.
- Exemplul este clasificat în clasa care are cel mai mare scor de încredere dintre toți clasificatorii.

MSV structurate

- Normală f: $\mathcal{X} \rightarrow \mathbf{R}$
  - Intrări de orice fel
  - Ieşiri numerice (naturale, întregi, reale)
- Structurată: $\mathcal{X} \rightarrow \mathcal{Y}$
  - Intrări de orice fel
  - Ieşiri de orice fel (simple sau structurate)

Curs 4: Rețele neuronale artificiale (RNA)

RNA detaliată în PDF-ul "Rețele neuronale artificiale (RNA)"!!!

Curs 5: Deep Learning

ANN

Layere:

- Input layer - size = input size (features)
- Hidden layer - various sizes (layers, neurons/layer)
- Output layer - size = output size (classes)

CNN

Mai multe layere (< 10)

Mai multe noduri per layer

Layere:

- Convolutional layer - feature map
- Pooling/Aggregation layer - size reduction
- Fully-Connected layer - answer

Output layer:

- Multiclass SVM:
    - Cel mai mare scor indică rezultatul corect
- Softmax (normalized exponential function):
    - Cea mai mare probabilitate indică rezultatul corect
    - Convertește scoruri în probabilități

- Sisteme care învaţă singure (SIS)
  - Reţele neuronale artificiale
    - Modele computaţionale inspirate de reţelele neuronale artificiale
    - Grafe speciale cu noduri aşezate pe straturi
      - Strat de intrare → citeşte datele de intrare ale problemei de rezolvat
      - Strat de ieşire → furnizează rezultate problemei date
      - Strat(uri) ascunse → efectuează calcule
    - Nodurile (neuronii)
      - Au intrări ponderate
      - Au funcţii de activare (liniare, sigmoidale, etc)
      - necesită antrenare → prin algoritmi precum:
        - Perceptron
        - Scădere după gradient
    - Algoritm de antrenare a întregii RNA → Backpropagation
      - Informaţia utilă se propagă înainte
      - Eroarea se propagă înapoi

Curs 7: Posibil invitat - Inteligență Artificială Generativă - LLMs

Curs 8: Posibil invitat - Reprezentarea numerică a datelor în Inteligența Artificială

Curs 9: Lanțuri Markov

Un lanț Markov este un model matematic care ne ajută să înțelegem și să prezicem comportamentul unui sistem care evoluează în timp.

sistem = o serie de stări posibile și de tranziții între aceste stări

Fiecare stare este o situație sau o condiție a sistemului într-un anumit moment în timp.

Proprietatea lui Markov:

"Starea viitoare a unui sistem stocastic este determinată exclusiv de starea sa actuală și nu depinde de întreaga istorie a sistemului."

Exemplu: aruncarea cu zarul

Ciclurile Markov pot fi reprezentate în diverse moduri în funcție de tipul de proces Markov și de nivelul de detaliu necesar în modelare:

- Graf orientat / diagramă de stări
- Arbore
- Matricea de tranziție

Curs 10: Reprezentări vectoriale pentru texte

Reprezentări rare (sparse):

- Mutual-Information weighted word co-occurrence matrices

Reprezentări dense (compacte):

- Singular Value Decomposition (și Latent Semantic Analysis)
- Neural-Network-Inspired models (skip-grams, CBOW)
- Altele (e.g. brown clusters)

Vectorii lungi = lungimea lor este între 20.000 și 50.000.
Vectorii rari = foarte multe elemente sunt 0.
Vectorii scurți = lungimea lor este între 200 și 1.000.
Vectorii denși = multe elemente nu sunt 0.
Vectorii denși pot generaliza mai bine, captând sinonimia termenilor.

Modele de predicție:

- Modelul Word2Vec
    - Skip-gram, CBOW (Continuous Bag of Words)
    - Se învață reprezentări, numite embeddings, ca parte din procesul de predicție/generare a textului.
    - Se antrenează o rețea neuronală pentru prezicerea următorului cuvânt

Curs 11: Rețele neuronale convolutive

Word embeddings + Transformers

Word embeddings:

- Static (context-free):
    - Word2Vec (2013)
    - GLoVe (2014)
- Dinamic (context-based):
    - ELMo (2018)
    - BERT (2019)

Arhitectură:

- Encoder:
    - Primește cuvinte
    - Construiește reprezentări
    - Modele bazate pe encoder:
        - Clasificare de propoziții
        - Clasificare de sentimente/emoții
        - Ex.: BERT

- Decoder:
  - Primește reprezentări (features) și alte input-uri
  - Generează secvențe de cuvinte
  - Modele bazate pe decoder:
    - Generare de texte
    - Ex.: GPT
- Modele bazate pe encoder-decoder (modele seq-to-seq):
  - Generare de texte care necesită un input (rezumat de text)
  - Ex.: BART, T5

Mecanismul de "atentie" (attention)

- Input = embedding-uri (de lungime n) pt v cuvinte – matrice v x n
- V – matrice v x v v – nr de cuvinte
- K – matrice n x d n – nr de features (lungimea unui embedding)
- Q – matrice n x d d – nr de features abstracte

Features

x_1	x_2	x_3	x_4
2	0	0	2
0	1	0	0
0	2	1	0
0	0	1	1
2	0	0	0
1	0	1	1

Transformare

input -> input x K x Q^T x $input^T$ x V x input -> features

(v,n) -> ((v,n) x (n,d)) x ((d,n) x (n,v)) x ((v,v

(v,n) -> (v,d) x (d,v) x

(v,n) -> (v,v) x

(v,n) -> (v,n)'

Scalare

input -> input x K x Q^T x $input^T$ x V x input -> fe

(v,n) -> ((v,n) x (n,d)) x ((d,n) x (n,v)) x ((v,v

(v,n) -> (v,d) x (d,v) x

(v,n) -> (v,v) / **√v** x

(v,n) -> (v,n)'

W_V

10	0	0	0	0	0
0	0	0	10	0	0
0	10	0	0	0	0

v_1	v_2	v_3	v_4
20	0	0	20
0	0	10	10
0	10	0	0

MatMul

.3	.3	.4	.1
.9	.3	.2	.7
.1	.1	0	.1
.3	.3	.4	.1

Attention Weight Matrix (A)

z_1	z_2	z_3	z_4

Attention Weighted Features

Normalizare

input -> input x K x Q^T x $input^T$ x V x input -> features

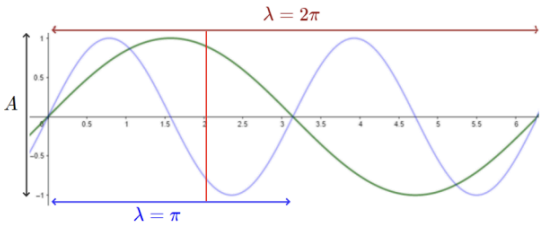
(v,n) -> ((v,n) x (n,d)) x ((d,n) x (n,v)) x ((v,v) x (v,n))

(v,n) -> (v,d) x (d,v) x (v,n)

(v,n) -> **softmax (**(v,v) / **√v)** x (v,n)

(v,n) -> (v,n)

Sin-based encoding



- Exemplu
  - O propoziție cu 5 cuvinte și d = 6
  - PE(cuv) = valorile functiei sin pentru diferite argumente (frecvențe sau lungimi de undă)
    - E.g. Sin(2 π pos / λ_i), i =0,1,2,..,d-1

pos		$\lambda = \pi$	$\lambda = 2\pi$	$\lambda = 3\pi$	$\lambda = 4\pi$	$\lambda = 5\pi$	$\lambda = 6\pi$
0	→	sin(0)	sin(0)	sin(0)	sin(0)	sin(0)	sin(0)
1	→	sin(2*1)	sin(1)	sin(2/3*1)	sin(2/4*1)	sin(2/5*1)	sin(2/6*1)
2	→	sin(2*2)	sin(2)	sin(2/3*2)	sin(2/4*2)	sin(2/5*2)	sin(2/6*2)
3	→	sin(2*3)	sin(3)	sin(2/3*3)	sin(2/4*3)	sin(2/5*3)	sin(2/6*3)
4	→	sin(2*4)	sin(4)	sin(2/3*4)	sin(2/4*4)	sin(2/5*4)	sin(2/6*4)

sim(PE(word$_0$),PE(word$_k$))=0!!!