Inteligență Artificială

Notare: 50% examen din teoria din cele două părți + 50% notă la laborator

Curs RC

Profesor: Hristea F.
Cursurile pe Drive

Ce este Inteligenţa Artificială?

Inteligenţa artificială poate fi privită ca fiind o ştiinţă sui-generis al cărei obiect de studiu îl constituie modul de programare a calculatoarelor în vederea săvârşirii unor operaţiuni pe care, deocamdată, oamenii le efectuează mai bine.

La inteligență artificială încercăm să căutăm soluția optimă (de exemplu, cea mai bună mutare la șah, sau următorul răspuns pe care l-ar da un program care vorbește cu utilizatorul).

Algoritmul A*

Video foarte bun de la Computerphile (nu prea am găsit de la indieni): A* (A Star) Search Algorithm - Computerphile - ține 14 minute

Este explicat cu animații și pseudocod pe Wikipedia: A* search algorithm

Algoritmul minimax și alfa-beta retezare

Video foarte bun (nu e de la indieni): Algorithms Explained – minimax and alpha-beta pruning - ține 11 minute

Rețele Bayesiene

- RețeleBayesiene.pdf
- Articolul Introduction to Bayesian Networks
- Videoul Understanding Bayesian networks and statistics (part2): Graphical models and applications
- Exemplu foarte simplu de pe Wikipedia

Curs ML

Profesor: Ionescu R.
Cursurile pe Drive
Modele de examen rezolvate

Noțiuni de bază

Pentru noțiuni de **algebră liniară**, **probabilități**, **statistică** sau referințe/formule la machine learning și neural networks: **cartea** Deep Learning Book de la MIT Press, disponibilă gratuit online.

Pentru **videouri** pe scurt și bine explicate despre orice legat de probabilități/statistică: **StatQuest** with Josh Starmer

Învățarea Automată

- **How Machines Learn** de la CGP Grey

Clasificatorul Bayes Naiv (Naive Bayes)

- Naive Bayes, Clearly Explained de la StatQuest

Măsurarea Performanței

Printre altele:

- Machine Learning Fundamentals: **Cross Validation** de la StatQuest
- Classification: **Accuracy** | Machine Learning Crash Course de la Google
- Classification: **Precision and Recall** | Machine Learning Crash Course de la Google
- Matricea de confuzie
- Receiver Operating Characteristic (**ROC**) și Area Under Curve (**AUC**)

Normalizare

- Normalizare **L1/L2**: Gentle Introduction to **Vector Norms** in Machine Learning
- Normalizare **min-max**: https://en.wikipedia.org/wiki/Feature_scaling#Rescaling_(min-max_normalization)
- Normalizare **standard** / standardizare: https://en.wikipedia.org/wiki/Feature_scaling#Standardization_(Z-score_Normalization)

Metoda Celor Mai Apropriați Vecini (k-Nearest Neighbors)

- StatQuest: K-nearest neighbors, Clearly Explained
- https://cs231n.github.io/classification/#nearest-neighbor-classifier

Blestemul Dimensionalității (Curse of Dimensionality)

Curse of Dimensionality Definition

Regresie Ridge (Ridge Regression)

O variantă a regresiei liniare la care se adaugă regularizare L2.
Descrisă foarte bine în documentația sklearn:
https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression-and-classification

Mașini cu Vectori Suport (Support Vector Machine)

- **Support Vector Machines**, Clearly Explained!!!
- Partea despre SVM din CS231n: https://cs231n.github.io/linear-classify/

Metode kernel (Kernel Methods)

- The **Kernel Trick**, parte a cursului Introduction to Computer Vision de la Udacity
- Sunt explicate și în video-ul de la SVM-uri.

Funcții de pierdere (Loss Functions)

- Loss Functions Explained de Siraj Raval
- Visual Information Theory care oferă o prezentare grafică a **cross-entropy**

Coborârea prin gradient (Gradient Descent)

- Gradient descent, how neural networks learn | Deep learning, chapter 2 de la 3Blue1Brown

Rețele Neuronale (Neural Networks)

- Articolul lui Karpathy **Hacker's guide to Neural Networks**
- Primul capitol din **Intro to Deep Learning with PyTorch**
- But what is a Neural Network? | Deep learning, chapter 1 de la 3Blue1Brown
- Perspectiva geometrică a rețelelor neuronale: Neural Networks, Manifolds, and Topology

Antrenarea rețelelor neuronale

- De ce trebuie să inițializezi random ponderile dintr-o rețea: Initializing neural networks - deeplearning.ai
- **Backpropagation**

Rețele Neuronale Convoluționale (Convolutional Neural Networks)

- **Conv Nets: A Modular Perspective** de pe blogul lui Christopher Olah

- Pentru a înțelege **stride**, **padding**, kernel **size**, etc: Intuitively Understanding Convolutions for Deep Learning
- An intuitive guide to Convolutional Neural Networks
- CS231n de la Stanford

Seminar RC

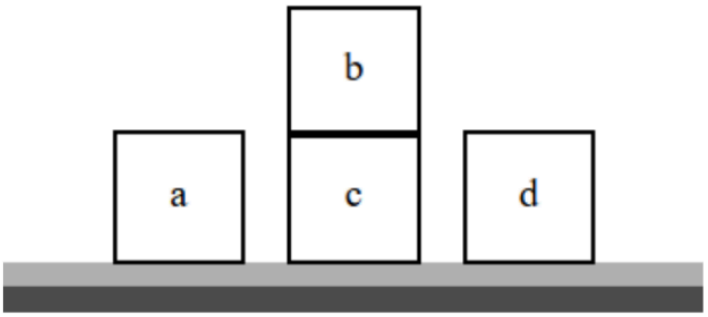
Probleme A*

Răspunsuri la întrebări legate de aplicarea algoritmului A*, în ordinea din document.

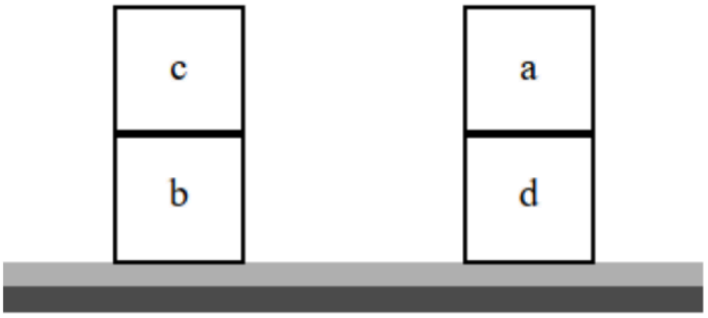
Problema blocurilor

1.
    a. Numărul de cuburi *M* și numărul de stive *N*; configurația finală și cea inițială a cuburilor.
    b. Configurația curentă a cuburilor și stivelor (ce cuburi sunt pe fiecare stivă).
2.
    a. Nodul de start: cuburile puse în configurația inițială.
    b. Nodul scop: cuburile puse în configurația finală.
3.
    a. O mutare înseamnă să luăm un cub din vârful unei stive și să îl punem în vârful altei stive.
    b. Pentru fiecare cub din vârful unei stive, avem *N-1* mutări posibile (presupunând că nu îl punem tot în aceeași stivă).
4. Putem considera că orice mutare are cost 1 (în acest fel, va căuta drumul cel mai scurt, adică soluția cu cele mai puține mutări).
5. Definim o euristică care estimează câte mutări avem de făcut să ducem toate blocurile în pozițiile lor finale:
    - dacă un bloc se află deja în poziția finală, are distanța 0
    - dacă stivele/blocurile ar forma o matrice, calculăm distanța Manhattan între poziția blocului și unde trebuie să ajungă (adică $|x_{final} - x_{inițial}| + |y_{final} - y_{inițial}|$)

Pe exemplul dat:



Exemplu de stare finala:



Pe acest exemplu, euristica în starea inițială:

- d este pe poziția corectă, adunăm 0 la distanță
- a e pe poziția (0, 0), ar trebui să fie pe (1, 2), deci adunăm |1 - 0| + |2 - 0| = 3
- b e pe poziția (1, 1), ar trebui să fie pe (0, 0), adunăm 2
- c trebuie să se mute de la (1, 0) la (0, 1), adunăm 2

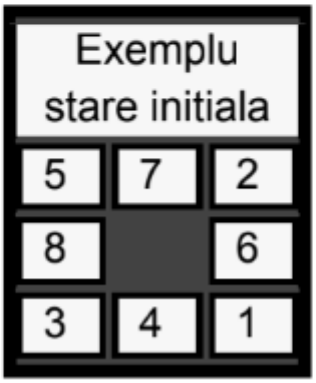
În total: 0 + 3 + 2 + 2 = 7

Problema 8-puzzle

1.
    a. Starea inițială și starea finală (numărul de tăblițe și de celule este hardcodat)
    b. Pozițiile curente ale tăblițelor.
2.
    a. Tăblițele pe pozițiile din starea inițială.
    b. Nodul scop este să fie tăblițele în configurația finală.
3.
    a. O mutare înseamnă că una dintre tăblițele de lângă spațiul liber sunt mutate în spațiul liber.
    b. Putem găsi spațiul liber, și vedem ce tăblițe pot fi mutate acolo.
4. Fiecare mutare are cost 1 (în felul ăsta o să încerce să găsească o rezolvare cu număr minim de mutări).
5. O euristică ar putea fi numărul de tăblițe care nu sunt la locul lor. Aceasta nu este o estimare foarte bună.

O euristică mult mai bună ar fi să luăm fiecare tăbliță și să calculăm distanța Manhattan față de poziția ei corectă, și să însumăm distanțele.

Exemplu:



Distanțele sunt:

- 5: distanță 2
- 7: distanță 3
- 2: distanță 1
- 8: distanță 2
- 6: distanță 0
- 3: distanță 4
- 4: distanță 2
- 1: distanță 4

Valoarea euristică pentru această stare inițială: 2 + 3 + 1 + 2 + 0 + 4 + 2 + 4 = 18

Problema canibalilor și misionarilor

1.
    a. Numărul de canibali/misionari *N*, și numărul de locuri în bărci *M*.
    b. Numărul de canibali/misionari de pe fiecare parte a râului (nr. de canibali pe o parte trebuie să fie <= nr. de misionari ca să fie o configurație validă). **Observație:** am putea să reținem doar numărul de canibali/misionari de pe o parte a râului (pentru că pe cealaltă se află *N - acest număr*)
2.
    a. Nodul de start: avem *N* canibali și *N* misionari pe malul de început, și 0 pe celălalt.

b. Nodul final: nu mai avem niciun canibal/misionar pe malul inițial, toți au trecut râul.

3.
   a. O mutare înseamnă că ducem sau aducem înapoi de pe un mal pe altul un număr de canibali și de misionari, astfel încât
      - numărul de canibali transportați pe barcă să fie <= numărul de misionari
      - numărul de canibali + numărul de misionari transportați să fie cel mult *M*
   b.
4. Fiecare mutare (călătorie cu barca) va avea cost 1.
5. O euristică ar putea fi numărul total de oameni pe malul inițial. Cu cât aducem mai mulți oameni pe celălalt mal, cu atât mai bine.
   Observație: O problemă care apare este că dacă o definim așa, funcția asta nu este monoton descrescătoare:
   - dacă transportăm *k* oameni o să avem *N - k* pe malul inițial
   - apoi cineva trebuie să aducă barca înapoi și o să avem *N - k + 1* pe mal
   - apoi scade iar

   Workaround: putem să facem
   floor((numărul de oameni pe malul inițial) / 2)
   În felul ăsta o să fie descrescătoare (omul care aduce barca nu mai afectează formula).

Laborator RC

Laborant: Maria Negru
Materiale laborator
Test de laborator (după cele 7 săptămâni): o să fie prin aprilie, trebuie să nu lipsim

Notare:
- test de laborator (80% din notă)
- rezolvarea la laborator a unor probleme mai dificile, după ce trecem de partea introductivă de Python (20% din notă)

Începând de la laboratorul din 11.03.2020, o să primim puncte bonus la laborator pentru activitate.

Laboratoare rezolvate