UNIVERSITATEA DE STAT DIN MOLDOVA
FACULTATEA DE MATEMATICÃ ȘI INFORMATICÃ
SPECIALITATEA IFORMATICÃ

Laboratorul 4

A efectuat: Tican Ion, grupa IA2201

A verificat: Valeriu Ungureanu, dr., conf. universitar

Chișinău 2024

Capitolul 1 Învățarea Automată

Învățarea automată (Machine Learning - ML) reprezintă un domeniu esențial al inteligenței artificiale, care permite sistemelor informatice să învețe din date și să își îmbunătățească performanța fără a fi programate explicit pentru fiecare sarcină. Această capacitate este realizată prin construirea și antrenarea modelelor matematice ce pot analiza date, identifica tipare și face predicții. Învățarea automată se bazează pe trei paradigme principale: învățarea supervizată, învățarea nesupervizată și învățarea prin întărire, fiecare dintre ele având aplicații distincte și metode specifice.

1. Învățarea supervizată

Învățarea supervizată se bazează pe utilizarea unui set de date etichetat pentru antrenarea modelelor. Fiecare intrare în setul de date este însoțită de o etichetă sau un răspuns care reprezintă rezultatul așteptat. Scopul modelului este de a învăța o funcție care poate asocia corect intrările necunoscute cu ieșirile corecte, folosind relațiile învățate din datele de antrenament.

Această metodă este utilizată pentru:
Clasificare: Determinarea unui rezultat dintr-un set finit de opțiuni (ex.: identificarea unui email ca "spam" sau "non-spam").
Regresie: Predicția unei valori continue (ex.: estimarea prețurilor locuințelor pe baza caracteristicilor lor).

Algoritmi populari în învățarea supervizată includ:
Regresia liniară și logistică, utilizate pentru predicții simple și clasificări binare.
Arborii de decizie și pădurile aleatoare (Random Forest), care sunt folosite pentru clasificări complexe și sarcini de regresie.
Mașinile cu suport vectorial (SVM), eficiente pentru clasificări în spații de mare dimensiune.
Rețelele neuronale, care sunt utilizate în aplicații avansate precum recunoașterea imaginilor și procesarea limbajului natural.

Un avantaj major al învățării supervizate este claritatea rezultatului datorită etichetării, dar aceasta necesită o cantitate mare de date etichetate, ceea ce poate fi costisitor.

2. Învățarea nesupervizată

Învățarea nesupervizată este utilizată pentru a analiza și descoperi tipare ascunse în date neetichetate. Spre deosebire de învățarea supervizată, nu există o ieșire specifică sau un răspuns corect asociat fiecărui exemplu. Scopul este de a înțelege structura datelor, gruparea lor sau relațiile dintre variabile.

Aplicațiile comune includ:
Clustering (grupare): Împărțirea datelor în grupuri similare (ex.: segmentarea clienților pe baza comportamentului lor de cumpărare).
Reducerea dimensionalității: Simplificarea datelor complexe prin păstrarea doar a celor mai relevante caracteristici, pentru vizualizare sau preprocesare (ex.: PCA - analiza componentelor principale).
Detectarea anomaliilor: Identificarea valorilor sau comportamentelor care se abat de la tiparele obișnuite (ex.: detectarea fraudelor bancare).

Algoritmii des utilizați includ:
K-means clustering, un algoritm simplu și eficient pentru gruparea datelor.
Algoritmi de grupare ierarhică, care construiesc o ierarhie de grupuri de date.
Modelele de amestec gaussian (Gaussian Mixture Models), care utilizează distribuții probabilistice pentru grupare.

Un avantaj al acestei paradigme este faptul că poate lucra cu date mari, neprocesate, însă rezultatele pot fi mai greu de interpretat.

3. Învățarea prin întărire (Reinforcement Learning - RL)

Învățarea prin întărire (Reinforcement Learning - RL) este o paradigmă diferită, bazată pe interacțiunea dintre un agent și un mediu. Agentul învață să ia decizii optime prin intermediul unui proces de încercare și eroare, obținând recompense pentru acțiunile corecte și penalități pentru cele greșite. Scopul este de a maximiza recompensa cumulată pe termen lung.

Această metodă este utilizată în:
Automatizare: Controlul vehiculelor autonome și al roboților industriali.
Gaming: Dezvoltarea agenților care pot învăța să joace jocuri complexe și să învingă jucători umani (ex.: AlphaGo).
Optimizarea resurselor: Managementul eficient al resurselor într-un mediu dinamic (ex.: managementul energiei sau al traficului aerian).

Algoritmii populari în RL includ:
Q-learning, un algoritm bazat pe învățarea valorilor asociate fiecărei acțiuni într-o anumită stare.
Deep Q-Networks (DQN), care utilizează rețele neuronale pentru a învăța politici optime într-un mediu complex.
Algoritmi Actor-Critic, care combină strategiile de politică și evaluare a stării pentru a optimiza deciziile.

Un exemplu practic al învățării prin întărire este utilizarea în robotică, unde un robot învață să se deplaseze într-un mediu necunoscut pentru a atinge un obiectiv, evitând obstacolele. În acest proces, robotul primește recompense pentru fiecare acțiune care îl apropie de obiectiv.

Capitolul 2 Învățarea Automată în limbajul Wolfram

Limbajul Wolfram oferă o platformă puternică pentru implementarea algoritmilor de învățare automată, datorită funcțiilor sale integrate și a suportului pentru analize complexe și vizualizări. Structura limbajului permite o abordare declarativă și simplificată, reducând complexitatea implementării algoritmilor. Cele trei paradigme majore ale învățării automate pot fi abordate eficient utilizând funcțiile Wolfram.

1. Învățarea supervizată în limbajul Wolfram

Pentru învățarea supervizată, Wolfram Mathematica oferă funcții integrate precum Classify și Predict. Acestea sunt concepute pentru sarcini de clasificare și regresie și pot încorpora mai multe tipuri de algoritmi fără a necesita codificarea manuală a acestora.

Funcția **Classify** permite crearea de modele pentru clasificarea datelor, alegând automat cel mai potrivit algoritm în funcție de datele furnizate. Exemple de utilizare includ clasificarea textului, a imaginilor sau a seturilor de date numerice.

Funcția **Predict** este utilizată pentru sarcini de regresie, cum ar fi estimarea valorilor viitoare pe baza unui set de date antrenat. Aceasta poate gestiona atât variabile numerice, cât și categorice.

Exemplu: Clasificarea mesajelor de spam cu ajutorul Classify

In[•]:=
```
data = {"Win a million dollars now!" → "Spam", "Meeting at 3 PM today" → "Not Spam",
   "Limited offer, click here!" → "Spam", "Your invoice is attached" → "Not Spam"};

classifier = Classify[data];
classifier["Win a hundred dollars now!"]
```

Out[•]= Spam

Exemplu: Predicția prețurilor la case cu ajutorul la Predict

In[•]:=
```
data = {<|"Size" → 1500, "Bedrooms" → 3|> → 300 000, <|"Size" → 2000, "Bedrooms" → 4|> →
     400 000, <|"Size" → 1800, "Bedrooms" → 3|> → 350 000};

predictor = Predict[data];
predictor[<|"Size" → 1600, "Bedrooms" → 3|>]
```

Out[•]= 317 445.

2. Învățarea nesupervizată în limbajul Wolfram

Învățarea nesupervizată este susținută prin funcții precum FindClusters și DimensionReduce, care sunt utilizate pentru analizarea și reducerea dimensiunii datelor.

Funcția **FindClusters** este un instrument puternic pentru analiza datelor neetichetate. Aceasta poate identifica grupuri naturale în date, aplicând algoritmi precum k-means, Gaussian Mixture

Models sau hierarchical clustering. Un exemplu de utilizare este segmentarea clienţilor pe baza comportamentului de cumpărare.

Funcţia **DimensionReduce** simplifică datele complexe păstrând caracteristicile esenţiale, fiind utilă pentru vizualizarea datelor în spaţii bidimensionale sau tridimensionale. Aceasta utilizează metode precum PCA (Principal Component Analysis) sau t-SNE (t-Distributed Stochastic Neighbor Embedding).

Aceste funcţii facilitează explorarea datelor şi extragerea caracteristicilor relevante, reducând timpul necesar implementării manuale a algoritmilor.

Exemplu: Funcţia FindClusters pentru a vedea datele financiare ale unui consumator

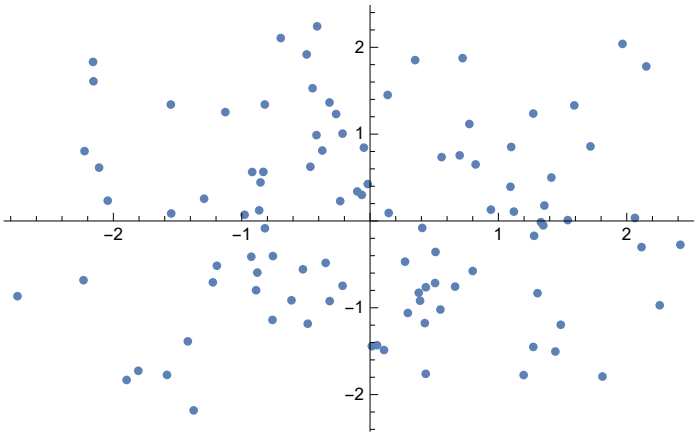
In[ ]:=
```
data = {{200, 3}, {150, 2}, {300, 5}, {100, 1}, {400, 6}, {250, 4}};
clusters = FindClusters[data];
clusters
```

Out[ ]=
```
{{{200, 3}, {150, 2}, {100, 1}, {250, 4}}, {{300, 5}, {400, 6}}}
```

Exemplu: Funcţia DimensionReduce pentru a vizualiza date dimensionale mari

In[ ]:=
```
data = RandomReal[1, {100, 5}];
reducedData = DimensionReduce[data, 2];
ListPlot[reducedData, PlotStyle → PointSize[Medium]]
```

Out[ ]=



3. Învăţarea prin întărire în limbajul Wolfram

Învăţarea prin întărire este mai puţin directă în limbajul Wolfram, dar poate fi implementată utilizând funcţii generice precum **Dynamic** pentru interacţiunea cu mediul şi **FindMaximum** pentru optimizarea funcţiei de recompensă. De asemenea, algoritmi personalizaţi pot fi construiţi utilizând bucle şi mecanisme de actualizare a stării.

Crearea unui agent RL poate implica simularea unui mediu folosind funcţii matematice sau stocastice şi utilizarea unor tehnici de optimizare pentru actualizarea politicilor.

În combinaţie cu biblioteci externe sau apeluri către API-uri personalizate, Wolfram Mathematica poate integra algoritmi de tip Q-learning sau Deep Q-Networks.

Deşi limbajul nu are funcţii dedicate învăţării prin întărire la fel de intuitive precum învăţarea supervizată sau nesupervizată, flexibilitatea sa matematică permite implementarea acestora într-

un mod creativ.

Exemplu: Folosind FindMaximum putem optimiza un caz obișnuit de recompense

In[•]:=
```
reward[state_, action_] :=
  Which[state == "A" && action == "Right", 10, state == "B" && action == "Left", 5, True, -1];
policy = Table[MaximalBy[{"Left", "Right"}, reward[state, #] &], {state, {"A", "B"}}];
policy
```

Out[•]=
```
{{Right}, {Left}}
```

Exemplu: Simularea unui mediu RL

In[•]:=
```
states = {"Start", "Middle", "End"};
actions = {"Forward", "Backward"};

transition[state_, action_] := If[state == "Start" && action == "Forward",
   "Middle", If[state == "Middle" && action == "Forward", "End", state]];
reward[state_, action_] := If[transition[state, action] == "End", 10, -1];

simulate[state_, steps_] := NestList[
   ({transition[#[[1]], "Forward"], reward[#[[1]], "Forward"]}) &, {state, 0}, steps];
simulate[
 "Start",
 5]
```

Out[•]=
```
{{Start, 0}, {Middle, -1}, {End, 10}, {End, 10}, {End, 10}, {End, 10}}
```

Exemplu: Implimentarea unui Q-Learning Algoritm

```
qTable = Association[{"Start", "Forward"} → 0, {"Middle", "Forward"} → 0];

updateQ[q_, s_, a_, r_, sNext_] :=
  q + 0.1 (r + 0.9 Max[Lookup[qTable, {sNext, #} & /@ actions, 0]] - q);

state = "Start";
action = "Forward";
reward = reward[state, action];
nextState = transition[state, action];

If[! KeyExistsQ[qTable, {state, action}], qTable[{state, action}] = 0];
qTable[{state, action}] =
  updateQ[qTable[{state, action}], state, action, reward, nextState];
qTable
```

Out[•]=
```
⟨|{Start, Forward} → 0.1 (0. + (-1) [Start, Forward]), {Middle, Forward} → 0|⟩
```

Capitolul 3

Crearea claselor pentru problema de clasificare multiclasă: Fizică, Biologie, Matematică, Istorie, Geografie, Chemie, Literatură.

In[•]:=
```
physics = TextSentences@WikipediaData["Physics"];
```

In[•]:=
```
biology = TextSentences@WikipediaData["Biology"];
```

In[•]:=
```
math = TextSentences@WikipediaData["Mathematics"];
```

```
history = TextSentences@WikipediaData["History"]
```

In[•]:=
```
geography = TextSentences@WikipediaData["Geography"]
```

In[•]:=
```
chemistry = TextSentences@WikipediaData["Chemistry"]
```

In[•]:=
```
literature = TextSentences@WikipediaData["Literature"]
```

Numărul de propozitii din fiecare clasă

In[•]:=
```
Length /@ {physics, biology, math, history, geography, chemistry, literature}
```

Out[•]=
```
{232, 354, 354, 255, 268, 302, 201}
```

Exemplu de frază din clasa history

In[•]:=
```
RandomChoice[history]
```

Out[•]=
```
Professor Charles Harding Firth, Oxford's Regius Professor of history in 1904
  ridiculed the system as best suited to produce superficial journalists.
```

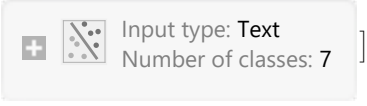
In[•]:=
```
topicadataset = Flatten[Thread /@ {physics → "Physics",
    biology → "Biology", math → "Mathematics", history → "History",
    geography → "Geography", chemistry → "Chemistry", literature → "Literature"}]
```

In[•]:=
```
RandomChoice[topicadataset, 4]
```

Out[•]=
```
{The earliest form of which there exists substantial knowledge is Greek drama. →
  Literature,
 However, in practice, mathematicians are typically grouped with scientists, and
    mathematics shares much in common with the physical sciences. → Mathematics,
 It can also have a social, psychological, spiritual, or political role. → Literature,
 Only one of them, the Riemann hypothesis, duplicates one of Hilbert's problems. →
  Mathematics}
```

Acum vom trece la antrenarea unui classifier

In[•]:=
```
topic = Classify[topicadataset]
```

Out[•]=
```
ClassifierFunction[ 
Input type: Text
Number of classes: 7 ]
```

Vom folosi acest classifier într-o propoziţie nouă

In[•]:=
```
topic["Stories common to a particular culture, but not
   supported by external sources (such as the tales surrounding King
   Arthur), are usually classified as cultural heritage or legends."]
```

Out[•]=
```
History
```

Vom folosi din nou acest classifier pentru a vedea probabilitatea

In[ ]:=
```
topic[
 "The Renaissance was a cultural movement that profoundly influenced European art",
 "Probabilities"]
```

Out[ ]=
```
⟨|Biology → 3.29679 × 10^-7, Chemistry → 0.0000250889,
Geography → 0.0051812, History → 0.527697, Literature → 0.465688,
Mathematics → 0.00137373, Physics → 0.0000347693|⟩
```

Vom vizuliza acum cum se schimbă probabilitatea în dependență de cuvinte pentru a înțelege cum lucrează acest classifier

In[ ]:=
```
visualizeSentence[input_] :=
 Module[{probabilities, cumulativeWords, styledWords, colorMapping, legend},
  colorMapping = <|"Physics" → Blue, "Biology" → Red, "Mathematics" → Purple, "History" →
     Green, "Geography" → Orange, "Chemistry" → Cyan, "Literature" → Yellow|>;
  cumulativeWords = Rest[FoldList[Append, {}, TextWords[input]]];
  probabilities =
   Quiet@Check[(topic[StringRiffle[#, " "], "Probabilities"] &) /@ cumulativeWords,
     Table[<||>, {Length[cumulativeWords]}]];
  styledWords = Table[With[{word = Last[cumulativeWords[[i]]],
      probs = probabilities[[i]]}, Style[word, Lookup[colorMapping,
       First@Keys@SortBy[probs, -# &], Black]]], {i, Length[cumulativeWords]}];
  legend = Grid[Table[{Style[topic, colorMapping[topic]], topic},
     {topic, Keys[colorMapping]}], Frame → True, Spacings → {1, 1}];
  Column[{Row[styledWords, " "], legend}]]

visualizeSentence[
 "Stories common to a particular culture, but not supported by external
   sources (such as the tales surrounding King Arthur), are
   usually classified as cultural heritage or legends."]
```

Stories common to a particular culture but not supported by external sources such as the tales surrounding King Arthur are usually classified as cultural heritage or legends

Out[ ]=

Physics	Physics
Biology	Biology
Mathematics	Mathematics
History	History
Geography	Geography
Chemistry	Chemistry
Literature	Literature

Putem observa că cuvântul „culture” are un impact pentru literatură, iar adăugând cuvintele ”King Arthur” acestea au crescut probabilitatea pentru History. Un alt exemplu

In[]:=
```
visualizeSentence[
 "The Renaissance was a cultural movement that profoundly influenced European art"]
```

The Renaissance was a cultural movement that profoundly influenced European art

Out[]=

Physics	Physics
Biology	Biology
Mathematics	Mathematics
History	History
Geography	Geography
Chemistry	Chemistry
Literature	Literature

In[]:=
```
topic["The oceans cover more than 70% of Earth's surface,
   regulating the climate, supporting diverse ecosystems, and providing
   food and resources for human populations.", "Probabilities"]
```

Out[]=
$$\langle |\text{Biology} \to 0.86479,\ \text{Chemistry} \to 5.922 \times 10^{-12},\ \text{Geography} \to 0.13521,\ \text{History} \to 6.28249 \times 10^{-8},\ \text{Literature} \to 8.37986 \times 10^{-10},\ \text{Mathematics} \to 2.86833 \times 10^{-9},\ \text{Physics} \to 3.63976 \times 10^{-10} | \rangle$$

In[]:=
```
visualizeSentence[
 "The oceans cover more than 70% of Earth's surface, regulating the climate,
   supporting diverse ecosystems, and providing
   food and resources for human populations."]
```

The oceans cover more than 70% of Earth's surface regulating the climate supporting diverse ecosystems and providing food and resources for human populations

Out[]=

Physics	Physics
Biology	Biology
Mathematics	Mathematics
History	History
Geography	Geography
Chemistry	Chemistry
Literature	Literature

In[]:=
```
Counts[topic[TextSentences[WikipediaData["Ancient Rome"]]]]
```

Out[]=
$$\langle |\text{Literature} \to 272,\ \text{History} \to 178,\ \text{Biology} \to 27,\ \text{Physics} \to 114,\ \text{Geography} \to 40,\ \text{Chemistry} \to 31,\ \text{Mathematics} \to 60 | \rangle$$

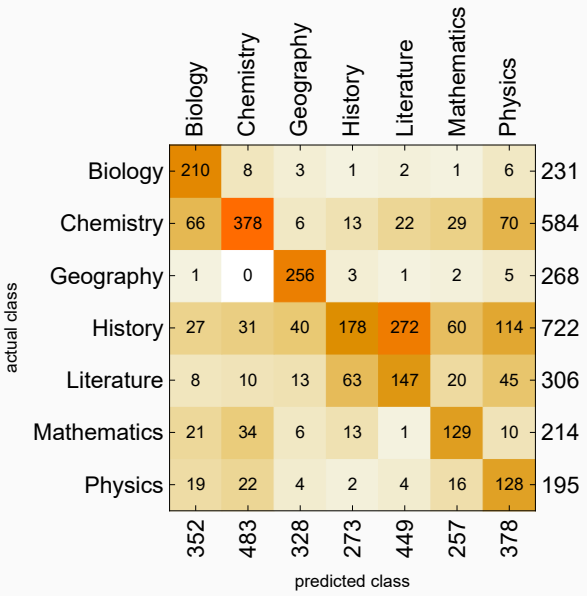
In[]:=
```
testset = Flatten[Thread /@ {TextSentences[WikipediaData["Cell (biology)"]] → "Biology",
      TextSentences[WikipediaData["Gravity"]] → "Physics",
      TextSentences[WikipediaData["Group theory"]] → "Mathematics",
      TextSentences[WikipediaData["Ancient Rome"]] → "History",
      TextSentences[WikipediaData["Geography of Earth"]] → "Geography",
      TextSentences[WikipediaData["Periodic table"]] → "Chemistry",
      TextSentences[WikipediaData["Shakespeare"]] → "Literature"}];
```

In[ ]:=
```
ClassifierMeasurements[topic, testset]["Report"]
```

Out[ ]=

Classifier Measurements

Number of test examples	2520
Accuracy	56.6% ± 0.99%
Accuracy baseline	28.7% ± 0.90%
Geometric mean of probabilities	0.0687 ± 0.0081
Mean cross entropy	2.68 ± 0.12
Single evaluation time	2.93 ms/example
Batch evaluation speed	4.76 examples/ms
Rejection rate	0%



In[ ]:=
```
Export["C:\\Users\\royal\\Documents\\topic.wmlf", topic]
```

Out[ ]=
```
C:\Users\royal\Documents\topic.wmlf
```

In[ ]:=
```
func = FormFunction[{"text" → "String"}, topic[#text] &];
```

In[ ]:=
```
FormFunction[
 DynamicModule[{Forms`Format`PackagePrivate`values},
  Dynamic[If[ValueQ[Forms`Format`PackagePrivate`values],
    Column[{Panel[Style[(topic[#text] &)[Forms`Format`PackagePrivate`va
        "Output", "StandardForm"], Background → White],
      Item[Row[{Spacer[0], Forms`Format`PackagePrivate`formCancelButtor
          "Back", Clear[Forms`Format`PackagePrivate`values],
          Grid[{{Item[Annotation[Style["text", "FormLabel"], "text",
                "HTMLLabel"], Alignment → Right], Annotation[InputField
                String, FieldHint → Null, DefaultBaseStyle → "FormField"
                DefaultFieldHintStyle → "FormFieldHint", FieldMasked →
                Enabled → Automatic], {"text", "data-field-name" → "Str
                "data-field-type" → "Structured", "data-field-verbose"
                 "string"}, "HTMLControl"]}, {"", ""}}, Alignment → Lef
            BaseStyle → {"ControlStyle", ShowStringCharacters → False}]
           AppearanceRules, "FormCancelButtonStyle"]]}], Alignment → Ri
     Alignment → Right], Forms`Format`PackagePrivate`values;
    Forms`PackageScope`bindForm[Forms`PackageScope`changeAppearanceRule
     Grid[{{Item[Annotation[Style["text", "FormLabel"], "text", "HTMLL
          Alignment → Right], Annotation[InputField["", String,
           FieldHint → Null, DefaultBaseStyle → "FormField",
           DefaultFieldHintStyle → "FormFieldHint", FieldMasked → False,
           Enabled → Automatic], {"text", "data-field-name" → "String",
           "data-field-type" → "Structured", "data-field-verbose" → "st
          "HTMLControl"]}, {"", ""}}, Alignment → Left, BaseStyle →
        {"ControlStyle", ShowStringCharacters → False}], "CancelLabel" →
     Forms`Format`PackagePrivate`values]], TrackedSymbols :→
    {Forms`Format`PackagePrivate`values}], DynamicModuleValues :→ {}]]
```

```
FormFunction[ ]
```

În urma testelor se atestă că procentajul de acurateţe este într-adevăr de 56%. Modelul dat răspunzând corect la date generale, dar la cele specifice poate fi confundat cu alte clase.