Ghid Complet: Inteligență Artificială și Învățare Automată

# 1. Ce este Inteligența Artificială (AI) și Învățarea Automată (ML)?

Inteligența Artificială (AI) este un domeniu al informaticii care se ocupă cu construirea de mașini sau programe capabile să "gândească" sau să ia decizii inteligente.

Învățarea Automată (Machine Learning - ML) este o ramură a AI care permite calculatoarelor să învețe din date, fără a fi programate explicit pentru fiecare sarcină.

Exemplu simplu: Dacă ai date cu numărul de ore de studiu și scorurile la test, ML poate învăța să prezică scorul pe baza orelor de studiu.

# 2. Tensorii

Un tensor este o structură de date care reprezintă un container de numere. Tensorii pot fi scalari (0D), vectori (1D), matrici (2D) sau tensori de ordin superior (3D, 4D etc.).

În PyTorch, tensorii sunt fundamentali pentru manipularea datelor și antrenarea modelelor.

Exemplu:

```python  
import torch  
x = torch.tensor([[1.0, 2.0], [3.0, 4.0]])  
print(x.shape) # (2, 2)  
```

# 3. Regresie (Regression)

Regresia este un tip de problemă de învățare automată în care modelul prezice o valoare continuă (nu o etichetă).

Exemplu de regresie: Prezicerea prețului unei case pe baza suprafeței în metri pătrați.

Cod simplu în PyTorch:

```python  
import torch  
import torch.nn as nn  
  
X = torch.tensor([[1.0], [2.0], [3.0], [4.0], [5.0]])  
y = torch.tensor([[50.0], [60.0], [70.0], [80.0], [90.0]])  
  
model = nn.Linear(1, 1)  
loss\_fn = nn.MSELoss()  
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)  
  
for step in range(500):  
 pred = model(X)  
 loss = loss\_fn(pred, y)  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
```

# 4. Clasificare (Classification)

Clasificarea este utilizată pentru a prezice etichete discrete sau categorii (ex: spam sau nu spam).

Modelul clasifică datele în funcție de caracteristicile lor.

Funcția de pierdere tipică: CrossEntropyLoss

Exemplu:

```python  
import torch.nn as nn  
output = torch.tensor([[2.0, 1.0]])  
target = torch.tensor([0])  
loss\_fn = nn.CrossEntropyLoss()  
loss = loss\_fn(output, target)  
print(loss)  
```

# 5. Funcții de pierdere (Loss Functions)

O funcție de pierdere măsoară cât de departe sunt predicțiile modelului față de valorile reale.

- MSELoss: pentru regresie.  
- CrossEntropyLoss: pentru clasificare.

Modelul încearcă să minimizeze această pierdere.

# 6. Optimizatori (Optimizers)

Optimizatorii actualizează greutățile modelului pentru a minimiza funcția de pierdere.

Exemplu:

```python  
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)  
```

Optimizatori comuni: SGD, Adam, RMSProp.

# 7. Rețele Neuronale (Neural Networks)

Rețelele neuronale sunt alcătuite din straturi de neuroni care procesează datele. Fiecare strat transformă intrările și transmite ieșirile mai departe.

Exemplu:

```python  
model = nn.Sequential(  
 nn.Linear(1, 10),  
 nn.ReLU(),  
 nn.Linear(10, 1)  
)  
```

# 8. Date de antrenament vs. test (Train vs. Test)

Datele sunt împărțite în două părți:  
- Date de antrenament: folosite pentru învățare  
- Date de test: folosite pentru a verifica cât de bine generalizează modelul.

Exemplu:

```python  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
```

# 9. Funcții de activare (Activation Functions)

Ele introduc non-linearitate în rețea și ajută la învățarea modelelor complexe.

Funcții comune: ReLU, Sigmoid, Tanh, Softmax

Exemplu:

```python  
import torch.nn.functional as F  
x = F.relu(x)  
```

# 10. Modele Clasice de ML

- Regressie Liniară

- Regressie Logistică

- Arbori de decizie (Decision Tree)

- Pădure Aleatoare (Random Forest)

- K-Nearest Neighbors (KNN)

- Mașini cu Vectori de Suport (SVM)

- Rețele Neuronale