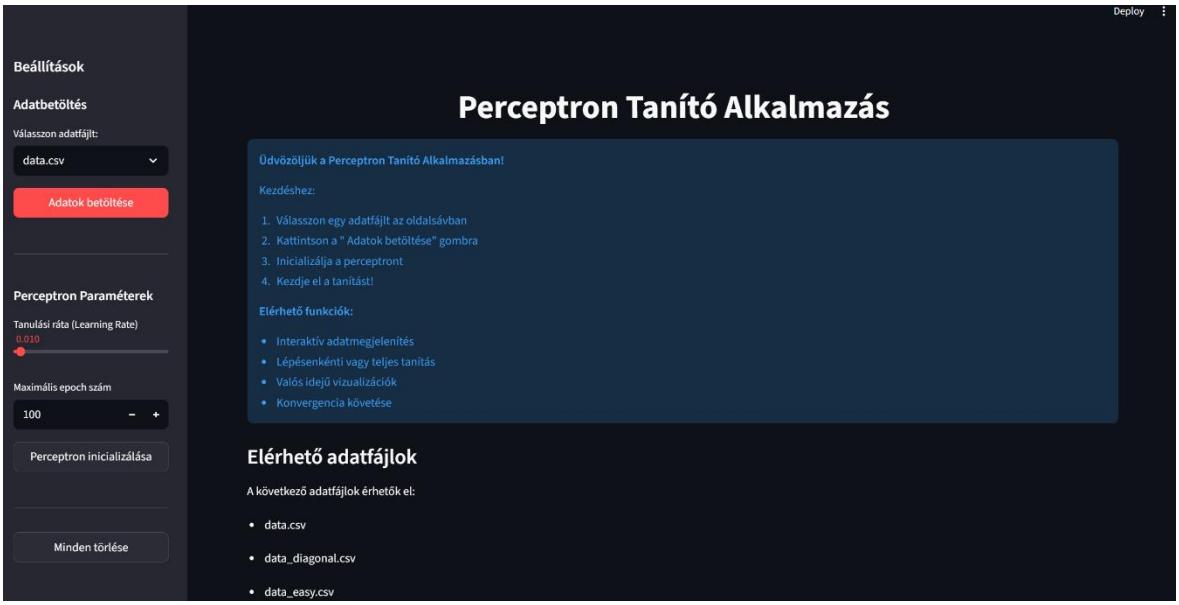
Haladó progamozás Interaktív percetron tanító

Radics Róbert Pál
JQ8R8D

2025

1. Bevezetés

Ez a dokumentáció egy perceptron alapú tanuló alkalmazást mutat be, amely egy egyszerű mesterséges intelligencia működését szemlélteti a gyakorlatban. A program célja az volt, hogy érthető módon bemutassa, hogyan képes egy algoritmus tanulni a megadott adatokból, és ezek alapján később önálló döntéseket hozni. Az alkalmazás elsősorban oktatási célokat szolgál, és segítséget nyújt a gépi tanulás alapjainak megértésében.



1.1. A projekt célja

A projekt célja egy olyan egyszerű program elkészítése volt, amely bemutatja a perceptron alapú tanulás működését. A cél nem egy bonyolult, professzionális mesterséges intelligencia létrehozása volt, hanem egy olyan alkalmazás fejlesztése, amely könnyen érthető, kipróbálható és jól szemlélteti az alap működési elveket. A felhasználó saját bemeneti adatokkal kísérletezhet, így közvetlenül megtapasztalhatja a tanulási folyamat lényegét

2. Rendszer áttekintésének működése

A program működése: a felhasználó CSV fájlból tölti be a kétdimenziós adatpontokat (x1, x2 koordináták) és azok osztálycímkéit (0 vagy 1). A perceptron kezdetben véletlenszerű súlyokkal és bias értékkel dolgozik. Minden adatpontra előrejelzést készít, amelyet összehasonlít a helyes címkével. Hibás predikció esetén a súlyokat és a bias-t módosítja a tanulási ráta és a hiba alapján. Ezt epoch-onként megismétli, így a rendszer egyre pontosabb válaszokat ad. Amikor a hibák száma nullára csökken, a perceptron konvergált. Ez a folyamat jelenti a tanulást.

```
x1,x2,label
0.5,4.5,1
1.0,4.0,1
0.3,4.7,1
0.8,4.2,1
0.6,4.4,1
1.2,3.8,1
0.2,4.8,1
1.4,3.6,1
0.5,4.5,1
0.9,4.1,1
0.4,4.6,1
0.7,4.3,1
0.6,4.4,1
1.1,3.9,1
0.3,4.7,1
1.3,3.7,1
0.5,4.5,1
0.8,4.2,1
0.2,4.8,1
0.7,4.3,1
0.6,4.4,1
1.0,4.0,1
0.4,4.6,1
1.4,3.6,1
0.5,4.5,1
4.5,0.5,0
5.0,1.0,0
4.3,0.3,0
```

2.1 Használati útmutató

2.1.1. Telepítés és előfeltételek

- Pythong csomag amely 3.8-as verziójú vagy újabb
    - python -m venv venv venv\scripts\activate
    - source venv/bin/activate
- Függőségek telepítésére szükség van, külöben a program, nem fog működni. Itt a requirements.txt hoztam létre, hogy a numpy, pandas, matplotlib, streamlit, plotly könyvtárakat egyszerűbben el lehessen érni.
    - pip install -r requirements.txt

2.1.2. Használat

- Webes felület: A webes felületnél, itt akár a program konzolján is képes egy paranccsal belépni vagy az erre létrehozott bat fájl.
    - streamlit run scripts/app.py
    - A mappában szereplő start.bat fájlra nyomva az oldalra is dob
- Működési sorrend a webes felületen:

1. Adatok betöltése: Oldalsávban válasszon CSV fájlt → "Adatok betöltése" gomb
2. Perceptron inicializálása: Állítsa be a tanulási rátát (0.01) és max epoch-ot (100) → "Perceptron inicializálása" gomb
3. Tanítás: "Tanítás" fül → "Egy lépés tanítás" vagy "Teljes tanítás" gomb
4. Eredmények: "Vizualizáció" fül (döntési határ) és "Konvergencia" fül (konvergencia görbe)

- Konzolos alkamazás: Itt egyedül a konzolos felület jelenik meg kiírásokkal, ahol az adott billentyű lenyomása után lehet a programot kezelni.

1. Program lefuttatása: python scripts/main.py
2. Menüvezérelt: adatbetöltés → inicializálás → tanítás →
3. vizualizáció.Saját adatok: Hozzon létre CSV fájlt a data/ mappában formátum: x1,x2,label (0 vagy 1 értékekkel

3. A Rendszer Architektúrája és Fő Komponensei

Az alkalmazás moduláris struktúrájú, ahol az adatfeldolgozás lépésről lépésre történik.**Komponens: AdatbetöltésSzerep**: A feldolgozandó forrás (CSV fájl) betöltése és validálása.**Implementáció**: DataLoader osztály (src/data_loader.py), Pandas könyvtár**Komponens: Perceptron AlgoritmusSzerep**: A bináris osztályozás végrehajtása súlyok és bias értékek használatával, valamint a tanítási folyamat kezelése.**Implementáció**: Perceptron osztály (src/perceptron.py), NumPy könyvtár**Komponens: VizualizációSzerep**: Az adatpontok, döntési határ és konvergencia görbe megjelenítése.**Implementáció**: Visualizer osztály (src/visualizer.py), Matplotlib könyvtár (konzolos), Plotly könyvtár (webes)**Komponens: Felhasználói FelületSzerep**: Interaktív vezérlés és valós idejű eredmények megjelenítése.**Implementáció**: Streamlit (scripts/app.py) webes felület, Python konzol (scripts/main.py)

3.1 Adatok betöltése: DataLoader

Az alkalmazás DataLoader osztályt használ a CSV fájlok betöltésére és validálására. A Pandas könyvtár segítségével olvassa be az adatokat, majd NumPy tömbökké konvertálja a feldolgozáshoz.

Bemenet: CSV fájl elérési útja

Kimenet:

X: NumPy tömb (N x 2), ahol N az adatpontok száma, 2 a jellemzők száma (x1, x2)

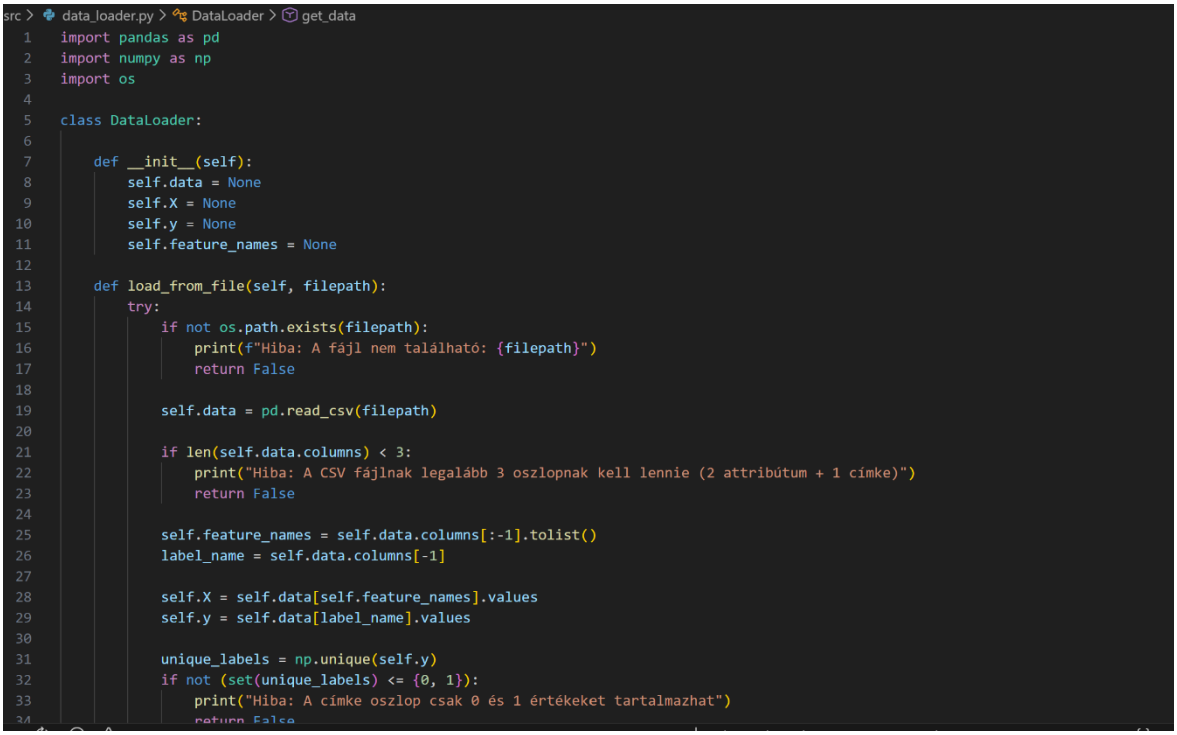
y: NumPy tömb (N,), osztálycímkék (0 vagy 1)

Validáció:

Ellenőrzi, hogy legalább 3 oszlop van-e (2 jellemző + 1 címke)

Ellenőrzi, hogy a címke oszlop csak 0 és 1 értékeket tartalmaz-e

Ellenőrzi, hogy legalább 2 bemeneti attribútum van-e



3.2 Perceptron algoritmus tanítási folyamata

A Perceptron osztály implementálja a klasszikus perceptron algoritmust. A tanítás során a program minden epoch után megmutatja, hogyan változnak a háttérben használt értékek (súlyok és bias). A felhasználó így nemcsak az eredményt látja, hanem a teljes tanulási folyamatot folyamatosan nyomon tudja követni.Inicializálás:

Súlyok (weights): Véletlenszerűen inicializálva [-0.5, 0.5] tartományban

Bias: Véletlenszerűen inicializálva [-0.5, 0.5] tartományban

Tanulási ráta (learning_rate): Felhasználó által beállított érték (alapértelmezett: 0.01)

Tanítási lépés (fit_step):

Minden adatpontra számítja az előrejelzést: prediction = activation_function(X · weights + bias)

Összehasonlítja a predikciót a valós címkével

Ha hiba van (prediction ≠ y_true):

Számítja a hibát: error = y_true - prediction

Frissíti a súlyokat: weights += learning_rate * error * X

Frissíti a bias-t: bias += learning_rate * error

Visszaadja az epoch hibáinak számát

Konvergencia: A kezdeti hibás válaszok száma fokozatosan csökken, miközben a rendszer egyre pontosabb működést ér el. Amikor a hibák száma nullára csökken, a perceptron konvergált.

```
Első 5 sor:
    x1   x2  label
0  2.5  3.2      1
1  1.8  2.1      1
2  3.1  4.0      1
3  2.0  2.5      1
4  2.7  3.5      1

Statisztikák:
              x1         x2      label
count  50.000000  50.000000  50.000000
mean    4.510000   2.358000   0.400000
std     1.941202   0.919869   0.494872
min     1.400000   0.700000   0.000000
25%     2.625000   1.700000   0.000000
50%     4.950000   2.300000   0.000000
75%     6.175000   3.000000   1.000000
max     7.400000   4.200000   1.000000
```

3.3 Döntési határ számítása

A perceptron döntési határa az a vonal, ahol a lineáris kimenet nulla: weights[0] * x1 + weights[1] * x2 + bias = 0Átrendezve: x2 = -(weights[0] * x1 + bias) / weights[1]A get_decision_boundary metódus 100 pontban számítja ki a döntési határt a megadott x1 tartományban, amelyet a vizualizációban zöld vonalként jelenít meg.

```
def get_decision_boundary(self, x_range):
    x1_min, x1_max = x_range
    x1_values = np.linspace(x1_min, x1_max, 100)

    if abs(self.weights[1]) < 1e-10:
        x1_boundary = -self.bias / self.weights[0] if abs(self.weights[0]) > 1e-10 else x1_min
        x2_values = np.linspace(x_range[0], x_range[1], 100)
        return np.full(100, x1_boundary), x2_values

    x2_values = -(self.weights[0] * x1_values + self.bias) / self.weights[1]

    return x1_values, x2_values
```

4. Tanítási Folyamat Részletes Elemzése

4.1. Epoch-onkénti Folyamat

Minden epoch során a perceptron végigmegy az összes adatponton:

1. **Előrejelzés**: Minden adatpontra kiszámítja a predikciót az aktuális súlyok és bias alapján
2. **Hibaszámítás**: Összehasonlítja a predikciót a valós címkével
3. **Súlyfrissítés**: Hibás predikció esetén módosítja a súlyokat és bias-t
4. **Hibák számlálása**: Megszámolja, hány adatpontot rosszul osztályozott
5. **Történet mentése**: Eltárolja a hibák számát és a súlyok/bias aktuális értékét

4.2. Konvergencia Feltételei

A perceptron konvergencia tétele szerint, ha az adatok lineárisan szeparálhatók, akkor a perceptron véges számú lépésben konvergál. A tanítás akkor fejeződik be, amikor:

- A hibák száma nullára csökken (konvergencia elérve)
- Vagy elérjük a maximális epoch számot

4.3. Paraméterek Hatása

- **Tanulási ráta (learning_rate)**:
- Kisebb érték (0.001-0.01): Lassabb, de stabilabb konvergencia
- Nagyobb érték (0.1+): Gyorsabb, de instabilabb lehet, oszcillálhat
- **Maximális epoch**: Ha az adatok nehezen szeparálhatók, több epoch szükséges

5. Vizualizáció és Eredmények

5.1. Adatpontok Megjelenítése

A Visualizer osztály Matplotlib (konzolos) vagy Plotly (webes) segítségével jeleníti meg az adatpontokat:

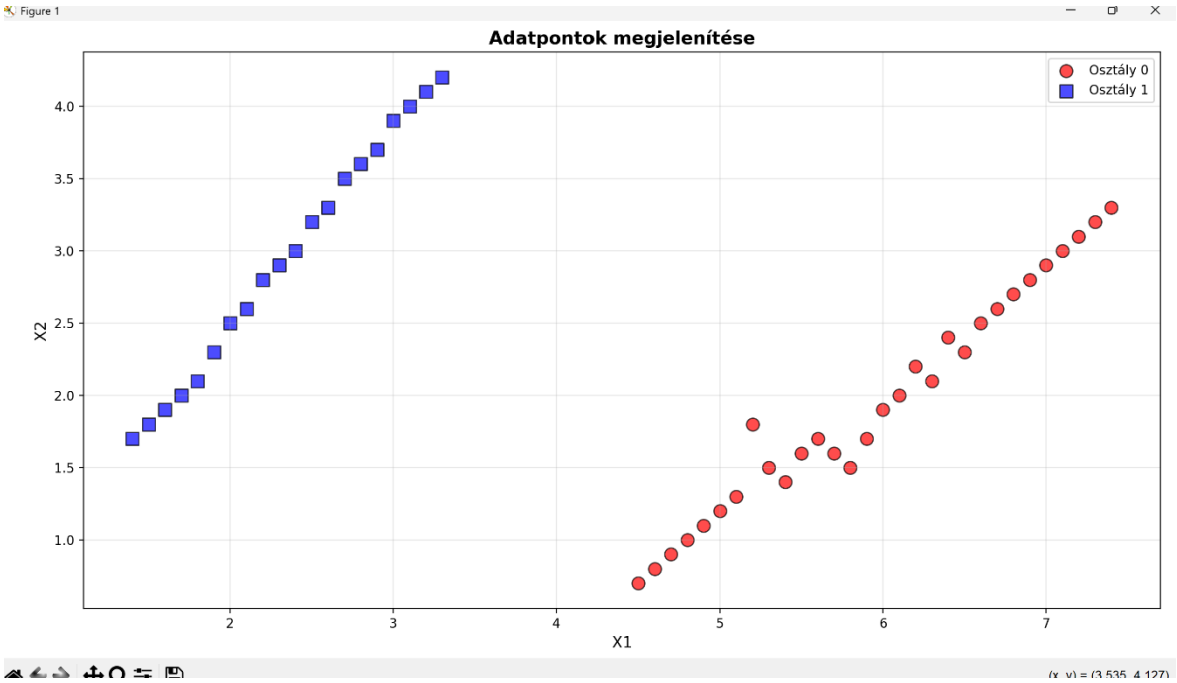
- Osztály 0: Piros pontok
- Osztály 1: Kék pontok
- Döntési határ: Zöld vonal

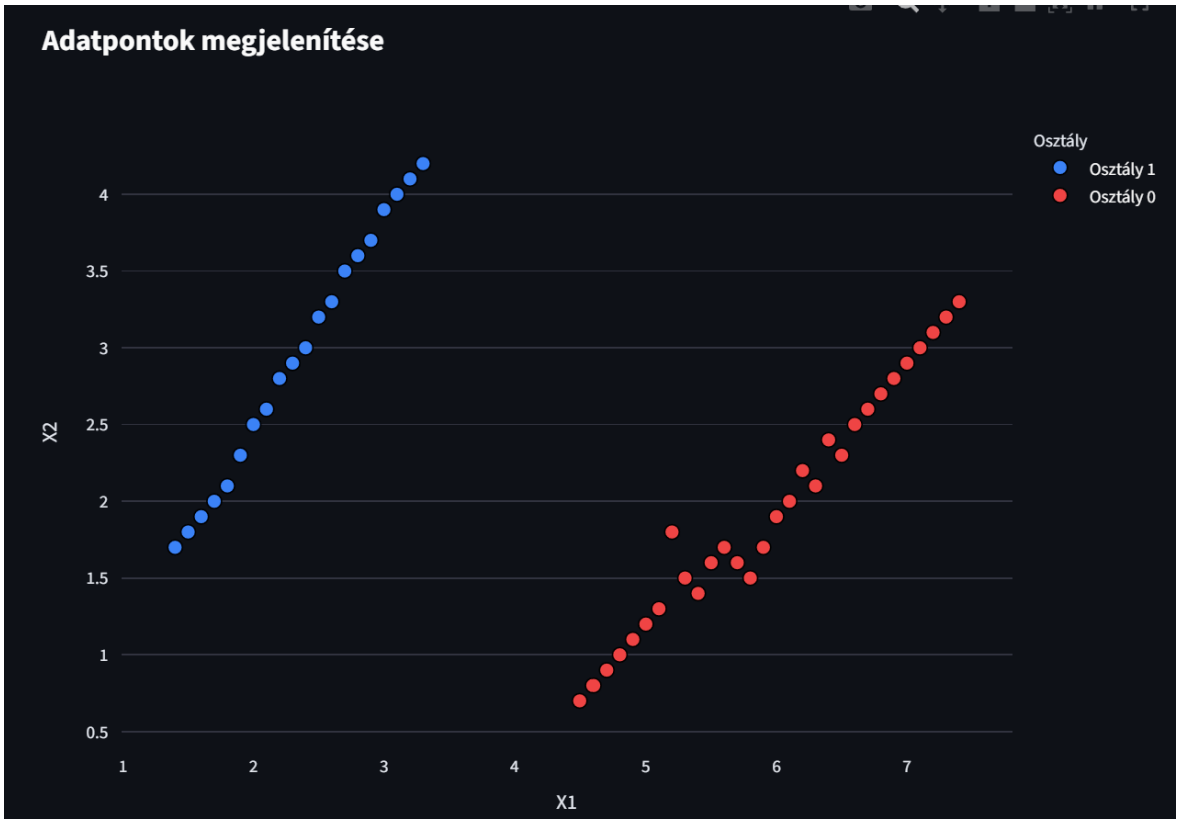
5.2. Konvergencia Görbe

A tanulási folyamat megjelenítése nemcsak számszerű formában jelenik meg, hanem grafikonon is nyomon követhető. A grafikon jól szemlélteti, hogy a kezdetben nagyobb hibaértékek fokozatosan csökkennek, ahogy a tanítás halad előre. Ez a vizuális megjelenítés nagyban segíti a tanulási folyamat megértését.[Ide szúrd be: 5. ábra – A tanulási folyamat grafikonja]

5.3. Epoch-onkénti Animáció

A webes felületen slider-rel választható, hogy melyik epoch állapotát nézze meg a felhasználó. Ez lehetővé teszi a döntési határ változásának lépésről lépésre való követését.




Adatpontok megjelenítése
Osztály
Osztály 1
Osztály 0
X2
X1
4
3.5
3
2.5
2
1.5
1
0.5
1
2
3
4
5
6
7

A kód elérhetősége és használt források

Elérhetőség:

https://github.com/RobertRadics/Percetron_App.git

Források:

https://docs.streamlit.io

https://numpy.org/doc/stable/user/index.html

https://plotly.com/python/