1. Bevezetés

A big data napjainkban az egyik vezető címke az informatikai terminológiák körében, különböző címekkel ellátva, mint: *Sikerhez és boldogsághoz vezet a big data* [?] vagy *Új korszak kezdődött a tudományban.* [?] De mit is jelent pontosan? Nincs explicit kimondott meghatározás a fogalomra, de Doug Laney 2001-es definíciója egy jó kindulópontnak tekinthető: az adatok nagy mennyiségben (volume), gyorsan(velocity) és különböző formátumban(variety) jelennek meg (3V's) [?].Azonban, ma már kiegészíthetjük ezt a fogalmat még 2V-vel: bizonyosság(veracity) és érték(value). [?] Az adatmennyiség amit előállítunk exponenciálisan növekszik olyan szintre, aminek tárolását, menedzselését és elemzését már nem tudjuk megoldani a saját, lokális erőforrásainkon belül az eddig megszokott adatelemzési eszközökkel, mint például Microsoft Excel, vagy különböző relációs adatbázis technológiák által. Becslések [?] szerint az adatok mennyisége kétévente duplázódik, így 2020-ra az összekézben forgó adatmennyiség elérheti a 44 zetabájtnyi (vagy 44 trillió gigabájtnyi) mennyiséget.

A "big data" lehetőséget biztosít arra, hogy ezeket az adatokat ne csak tároljuk, hanem új módokon tanuljunk belőle, értéket állítsunk elő, többet megtudjunk ügyfeleinkről, a saját üzleti folyamatainkról, ami versenyelőnyhöz vezethet. E mellett az áttörő kutatások számát is megnöveli azáltal, hogy rejtett összefüggéseket mutat meg. [?]

A cloud computing, és új technológiák megszületése és az, hogy a fizikai világ egyre jobban átterelődik az online térbe, új nehézségeket állít elő mind az adatokat kiszolgáló, mind az adatokat elemző infrastruktúrák számára. Ezek a problémák komoly gondot jelentek az informatikai iparnak, mivel érintik az fizikai manifesztációt (hardver), mind az ezt vezérlő és feldolgozó réteget (szoftver és algoritmus). Ezek a problémák, [?] –amelyek a tradícionális adattárház technológiákra jellemzőek– többek között származhatnak a hiba-tolerancia hiányából, a sokféle adatfajtából,a párhuzamosság hiányából, mely azt eredményezi, hogy a mai technológia fejlettség (és a központi számítási egységek fizikailag limitáltsága miatt) nem lesz megfelelő számítási teljesítmény a megnövekedett adatmennyiség menedzselésére.

2. A dolgozat célja

A technológia fejlődése és a számítási teljesítmény megnövekedése hozta létre azt az üzleti igényt [?], hogy egyre gyorsabban, egyre nagyobb adatmennyiség feldolgozása történjen meg. Ilyen igény például:csalás felderítés [?],

"dolgok" internete (IoT) [?] vagy alkalmazás monitoring [?]. Ez az adatfeldolgozási sebesség olyan szintre eljutott, hogy közel valós időben, az adat keletkezése után megtörténhet ennek feldolgozása. Ilyen gyorsaságú adatfeldolgozásra csak elosztott rendszerek segítségével vagyunk képesek, [?] amelyek felépítésükből fakadóan sok lehetőség és költség jellemez, amelyeket a későbbiekben fogok kifejteni. A dolgozatomban használt Apache Flink (mely az Apache Foundation egyik legújabb és legmodernebb terméke) platform közel 40 millió elem feldolgozására képes egy 40 magos architektúrán másodpercenként. [?].

Ahhoz, hogy ezt az adatmennyiséget ki tudjuk elemezni és ajánlásokat tudjunk adni, gépi tanulásra van szükségünk. A gépi tanulás az informatikának és a matematikának egy olyan ága, amely az adatok folyamatos betáplálása során új ismereteket szolgáltat, megpróbál előrejelzéseket adni anélkül, hogy explicit módon be lenne erre programozva. [?]. A választott metódus a stochastic gradient descent (SGD, sztochasztikus gradiens ajánlás) [?], amely egy olyan egyszerűsítési illetve optimalizációs eljárás, ahol adott célfüggvény gradiensét folyamatosan, iteratív módon számoljuk ki. Célom az, hogy megtervezzem Apache Flink alatt az SGD algoritmust, összehasonlítsam a teljesítményét a már implementált algoritmusokkal és megkezdjem a szükséges módosítások implementálást.

3. Gépi tanulás

Amikor tanulunk, a célünk, hogy minél jobb eredményeket érjünk el a számonkérésen, vagy minél több tudást halmozzunk fel, amit a későbbiek során (valószínűsíthetően) hasznosítani tudunk. A gépi tanulásnak is ugyanez a célja, különböző modellek megalkotása után a megadott példákból (input adat) különböző kimeneteket (output adat) ad ki. Az input adatokból próbál általánosítani oly módon, hogy az felhasználható legyen számára ismeretlen problémák során. Gépi tanulást használunk például:

- web keresés
- spam szűrés
- ajánló rendszerek
- online hírdetések

esetén is. Egy 2011-es Mckinsey riport [?] szerint a gépi tanulás (illetve a prediktív analitika) lesz a következő évek innovációinak alapja. IBM Watson-ja [?], már képes a beadott tünetek alapján, megjósolni, hogy mi

lehet a páciens betegsége (egyelőre még csak fejlesztőknek API-n keresztül).

Általánosan fogalmazva, az adat amit betáplálunk a gépi tanulás modellünkbe, tréning példáknak (training set) nevezzük. A tréning példák x, y párokat tartalmaznak, ahol x az érték vektor (feature vector). Minden x érték: kategórikus (diszkrét értékeksorozatból származik, plkék, piros, sárga) vagy numerikus (az érték egész vagy valós szám). y a címke (label), ami kategorizáló érték x-re nézve.A célünk az, hogy felfedezzük azt az

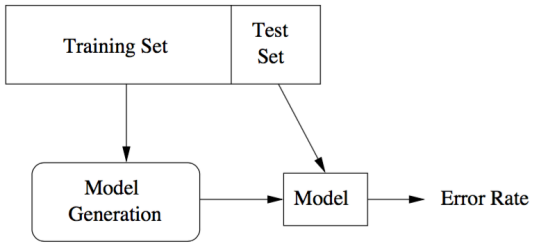
$$y = f(x)$$

függvényt, ahol a legjobban előre tudjuk jelezni az y értéket a meghatározott x-re nézve.

Két fő csoportja van a gépi tanulási algoritmusoknak: a felügyelt (supervised) és nem felügyelt (unsupervised) tanítás.

4. Felügyelt tanítás

Fontos, hogy szétválasszuk az adatainkat tréning és teszt adatokra. Ez biztosítja azt, hogy ne fordulhasson elő az a probléma, hogy a modellünk túlságosan fontos súllyal vesz egyes objektumokat az adatsoron (amik nem jellemzőek a lehetséges valós adatokra), ami azt eredményezi, hogy a valós problémákon már nem fog eredményesen működni. A problémát túltanulásnak vagy magolásnak (overfitting) nevezik.



1. ábra. Gépi tanulás általános modellje

5. Nem felügyelt tanítás

Nem felügyelt tanítás esetén adottak: $(x_1, x_2, ..., x_n)$ adataink, és nincs célfüggvényünk, vagy elvárt kimenetünk. Alapvetően nem struktúrált *zajból*

próbálunk mintázatot keresni, olyan modellt létrehozni, ami jól reprezentálja adatok valószínűségi eloszlálást. Annak ellenére, hogy nincs információnk arról, hogy az egyes adatok milyen kapcsolatban vannak egymással, (x_t) valószínűségi eloszlását meg tudjuk jósolni $(x_1, x_2, ..., x_{t-1})$ alapján, ahol $P(x_t|x_1, x_2, ..., x_{t-1})$ Egyszerűbb esetekben, ahol az input sorrend irreleváns, lehet modellt építeni az adatra, ahol$(x_1, x_2, ...)$ az adatsorunk, és ezek függetlenül de identically származnak a $P(x)^2$

6. Ajánlórendszerek

7. Batch

8. Lambda-architektúra

9. Streaming

bounded, unbounded

10. Time-agonistic

- Event time
- Proccesing time

11. Flink(Spark/Storm/Mapreduce)

12. Ajánlórendszerek

- Collaborative filtering
- Content based

13. ALS

14. DSG

15. Java, Scala

Max 1 oldal, miért ez lett

16. Dcg,nDCG