Farkas Dániel Szakmai házifeladat:

Machine Learning Engineer feladat

Első feladat (gyakorlati):

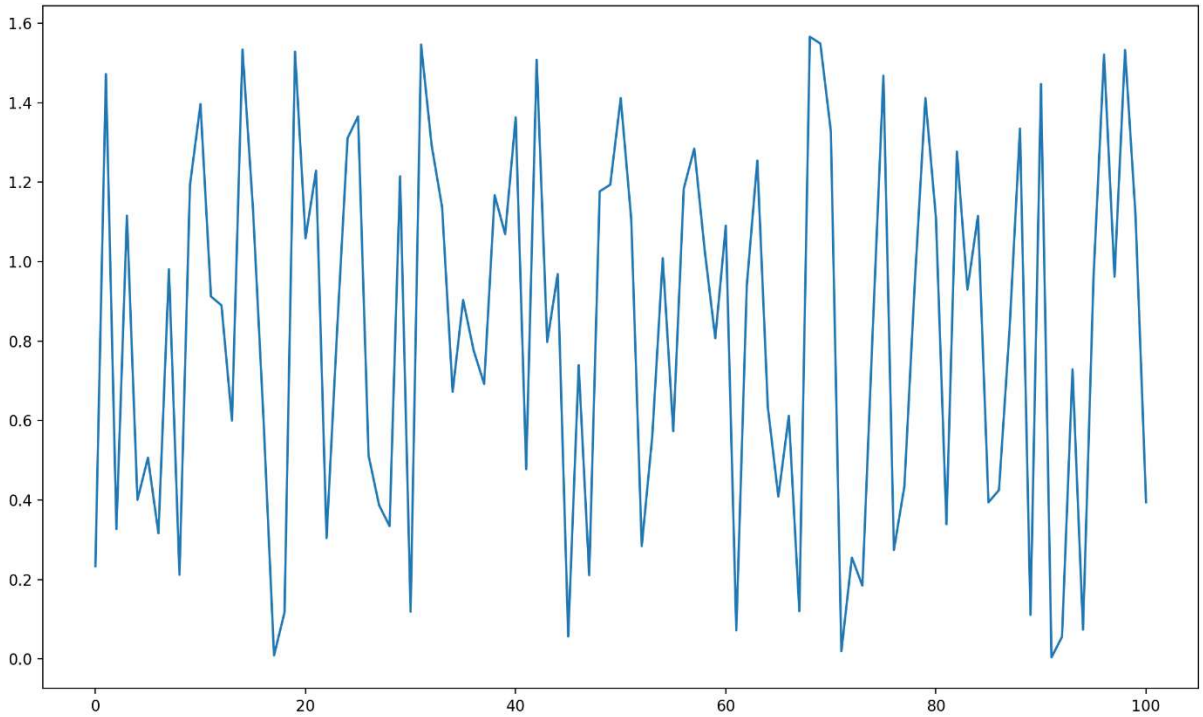
Mellékelten küldünk egy szöveges állományt, amely egy számsort tartalmaz. A feladat egy olyan neurális hálózat elkészítése, mely képes prediktálni a sorozat n. elemét. Ehhez a korábbi (megadott) értékek felhasználása megengedett.

A feladat kimenete:

- A megoldás menetének szöveges összefoglalása, dokumentációja
- A megoldáshoz használt forráskódok
- A tanítás során rögzített metrikák és ábrák magyarázattal
- A hálózatot futtató telepíthető bináris (x86_64 linux és windows)
  Bemenete: n egész szám
  Kimenete: Xn+1

Adatok értelmezése:

Az adatokon jól látszik, hogy oszcillálóak. tehát lineáris modellel nem lesz betanulható a mintázat. Az is látszik, hogy a file hatalmas (1.200.002 számot tartalmaz) így egyszerre nem beolvashatóak az adatok a memóriába.



Az ábrán az első 100 szám látható vizualizálva. Látható, hogy már itt megközelíti az 16 milliós értéket, azonban legmagasabb értéke is csak 15673249. Az is látható, hogy nem egy magától értetődő, egyszerű függvény határozza meg a számokat.

Architeaktúrális döntések:

A használt modell LSTM (long short-term memory). Ezt kifejezetten ajánlják sor predikciós problémáknál, hiszen a sorrendi összefüggéseket is képes kezelni, mivel rekurrens (vannak visszacsatoló kapcsolatai).

A hiba számítására MSE (mean squared error)-t használok.

A projekt Pytorch-ot használ NumPy-jal.

Négy csoportba sorolhatók a file-ok, ezek:

- **data_processing:** Az adatok beolvasásával, és feldolgozásával foglalkozó file-ok.
- **neural_network:** A Modelleket, és az ezeket használni képes hálózatot tartalmazza.
- **visualizer:** A feladat kiírás alapján nincs szükség ilyen file-ra, azonban hasznos riport-ok készítéséhez, ami bővítésnél hasznos lehet, és ez van használva a dokumentáció írásakor is.
- **src:** Nem tartalmaz forráskódot, az adat file-t és a teszt adat file-okat tartalmazza.

A négy csoporton kívül van egyetlen main file, ez fogja össze a projektet, határozza meg a futását.

Csoportok tartalma:

data_processing:

data_reader: Ez az osztály képes egy file-ból egyesével kiolvasni számokat. adatstream-ként is lehetséges számolni kis bővítéssel, jelenleg egyszerre meadott számú adatot olvas, és listaként továbbadja feldolgozni. Később képes folytatni a kiolvasást, nem kell elölről kezdeni.

sequence_dataset: Egy a torch által használható dataset. Konstruktora átvesz egy data_reader-t, abból képes később összeállítani saját adatrendszerét.

neural_network:

fc_model: Lineáris nn model, főleg tesztelésre volt hasznos, a kapott adatok feldolgozására nem célszerű használni.

lstm_model: A kapott adatok feldolgozására szánt nn modell.

neural_network: Megkapja a modellt, és megvalósítja hozzá a szükséges nn funkcionalitást (ezzel tudunk prediktálni, tanítani, és hasonlók. Itt van meghatározva a Loss function is.) alkalmazás bővítése esetén célszerű lehet általánosítani, de számsorok predikciójához ideális.

visualizer:

visualizer: Egyszerűen megkap egy adatsort, majd sorrend és érték alapján diagrammot készít belőle.

src:

data.txt: A kapott adat file.

test_data: Egy mappa, benne a data.txt-hez hasonló adatforrásokkal. Tesztelésnél hasznos megnézni, hogy például lineáris adatsoron jól működik-e az nn.

NN → neurális hálózat (neural network)*

Tanítás módja:

A háló adott számú (a teszteknél 100db) előző számból tanulja meghatározni a következő egyet. A tanítás a beérkező adatokon egy csúszóablakban történik. Először az első 100-ból predict-elek a 101-re, és ezzel tanítok. Majd a másodiktól a 101-edikig, és ebből cél a 102-et predict-elni. Majd így tovább.
Egy LSTM modellen tanítok, aminek az inputjai az előző adott mennyiségű szám. A modell kétirányú. A hidden node-ok, és a layer-ek száma megadható, 256 hidden node-al, és 4 layerrel történtek a tesztek. Majd egy egyszerű lineáris layerrel redukálom a node-okat, az egyetlen kimenetre.
A tanítás több epoch-al történik.

Az adatokat folyamatosan olvasom ki a file-ból majd ezekkel tanítok. Közben adott adatmennyiségenként végzek egy tesztet, olyan adatokkal, amikkel még nem volt tanítva a rendszer. A kapott eredményt kiírom konzolra, (hiszet output-on csak a végeredmény lehet.) majd ezekkel is tanítom a modellt. Amikor az adathalmaz végére ér, a program predikciót ad.

Pontosság mérése:

A pontosság mérés megtévesztő lehet, hiszen két szám összehasonlítására pontosság szempontjából csak akkor lehetséges objektívan, ha normalizáljuk őket. Ahhoz pedig meg kell határozni a szélsőértékeket, ami ismeretlen számsornál lehetetlen. Az adat scale-elve van, de 0-1 intervallumra normalizálni nem lehet. Így a hasonlóság relatív, viszont javulása objektíven meghatározható, és 0-100%-on mozog.

Használat:

A program használatához a bemenő adat file-jának helyét konzolosan kéri be a program. Majd bekéri az alap adatokat a háló felkészítéséhez.
Ezután el is kezdi a feldolgozást. A Konzol ablak adott tanítási ciklusszámonként kiírja az akkori tippelések hozzávetőleges pontosságát. (Ez hosszabb tanításnál hasznos, hiszen tudhatjuk, hogy még fut, és nagyjából hogy áll a feldolgozás.) Az adatfile végéhez érve, a futtatott file-al egy mappába helyezi a „prediction.txt” file-t, ami egyetlen integer számot tartalmaz, a predikciót egészre kerekítve. (Mivel a bemenetünk egész számokból áll, célszerűnek láttam tartani a formátumot.)

Tesztek:

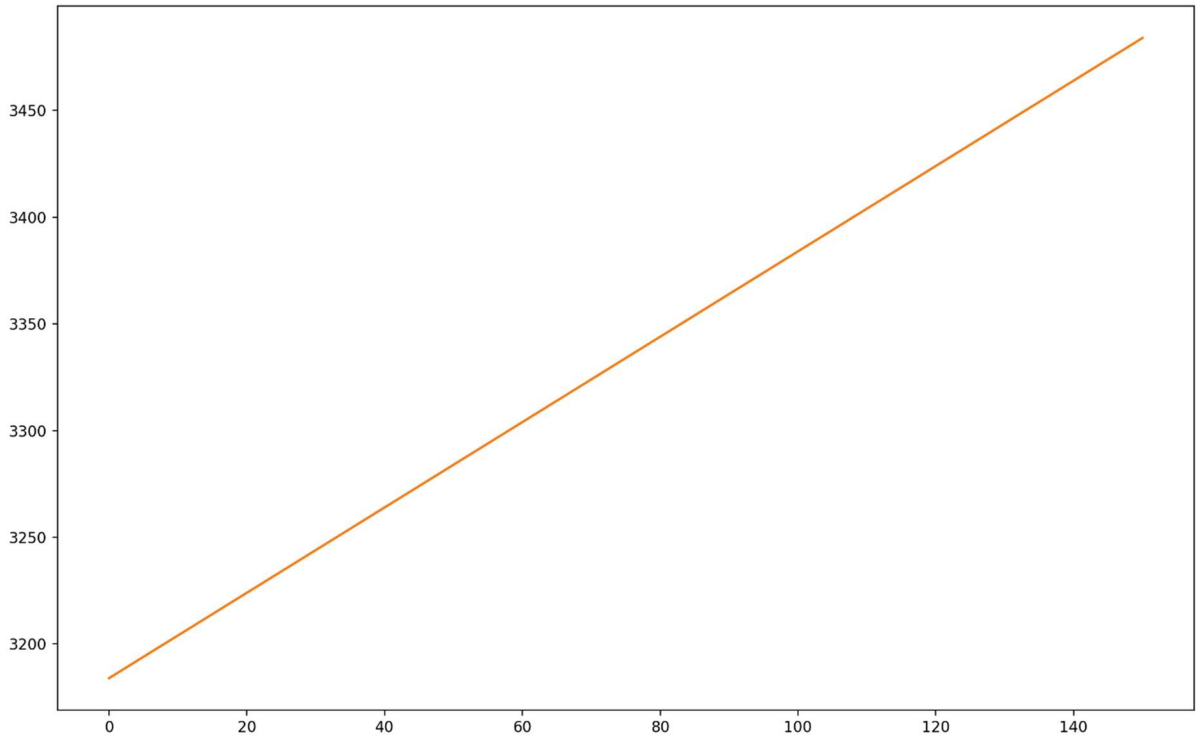
Környezeti:

Hogy kézzel fogható legyen, és látszódjon, hogy nem a hangolással van a baj, készítettem egy egyszerű lineáris számsorokat tanuló modellt, és egy egyszerű hozzá tartozó adat file-t (src/test_data/additive_test_data.txt). Ezzel a fennállással, rövidebb idő alatt is tesztelni lehetett olyan dolgokat, mint a tanítás, a normalizálás, az adatok megfelelő beolvasása, és ehhez hasonlók. Ezzel elméletileg tesztelhető a rendszer legtöbb eleme. Kivétel ez alól:

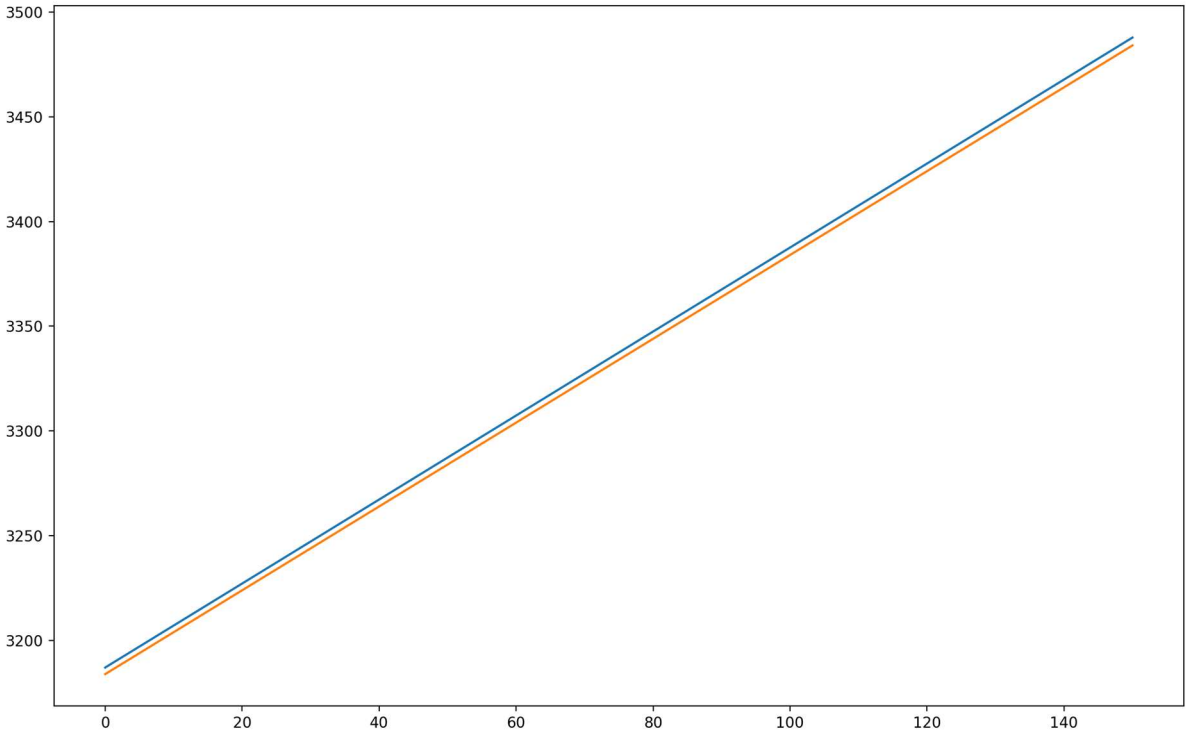
A modell ideális imput mérete, a learning rate. a batch méret, az epoch-ok száma, és maga a modell.

Természetesen a tanítási módszerre, és a loss function-re is reagálhat máshogyan két különböző modell, de ezekre lehet ajánlásokat keresni problémaspecifikusan.

A lineáris tanulásra tökéletesen hangolható volt a rendszer.



Az ábrán két egymást teljesen takaró egyenes látható. Az egyértelműség kedvéért alább beszúrok egy kicsit rosszabbul tanítottat is.

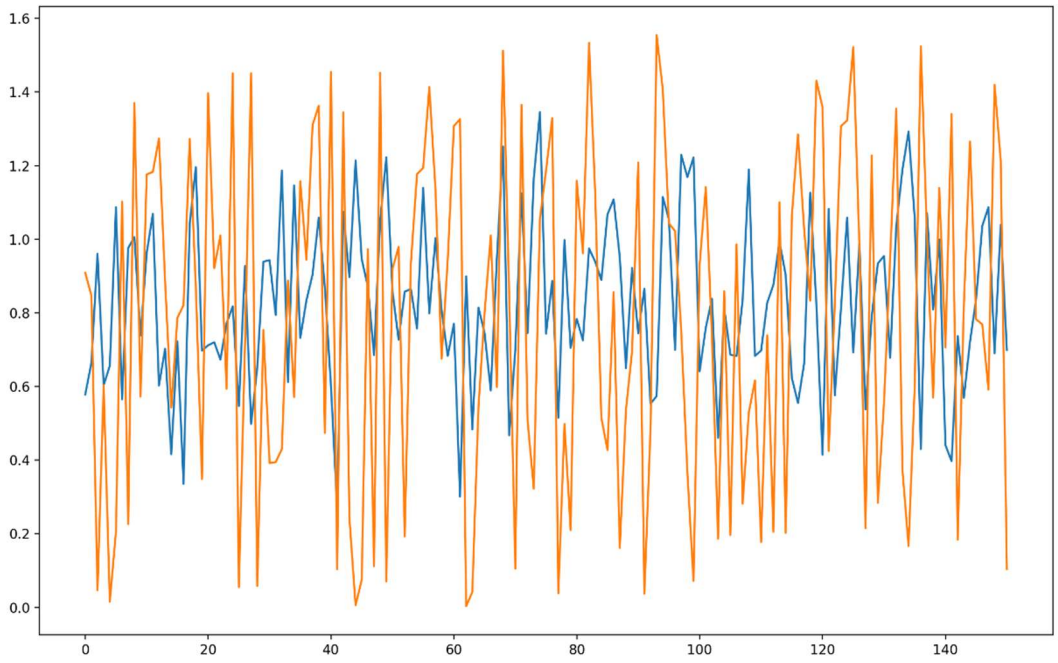


Ebből kiderül, hogy a környezet megfelelő beállításokkal alkalmas mesterséges inteligencia betanítására, és következő értékek jóslására.

A jóslás tesztelésekor egészre pontosan meg tudta határozni a sor következő elemét. Kerekítve megfelelő értéket ad.

Tesztek az adatokon:

Sajnos nem állt rendelkezésemre GPU-val rendelkező eszköz, így a tanítási folyamatok nagyon lassúak voltak, inkább kis adatmennyiségen futtattam teszteket. De kis adatmennyiségeknél is látszódik a predikció javulása.



Ezen az ábrán a narancssárga a valós, a kék pedig a tippelt érték. Látszik, hogy tompán, de nagyjából követi a mintát. Ennek eléréséhez 150 alkalommal volt 50-es input size-al tanítva 5 epoch-kal a háló, és egy alkalommal 256 számot olvastam ki. Tehát az adatokbó nagyjából 39.000 volt felhasználva a tanításhoz.

A modell tesztjei:

Az LSTM modell is volt tesztelve a lineáris adathalmazon. itt kifejezetten kis adatmennyiségen, azonban egész pontatlan eredményekre vezetett. Ez jelentheti a modell valamilyen hibáját, de azt is, hogy egyszerűen ennek a modellnek több tanulásra van szüksége, vagy azt, hogy valamilyen érték (például drop_out, vagy bias) rosszul van megadva.

Tesztek összefoglalása:

Valószínűleg valamilyen kisebb hiba lehet az LSTM-modellem beállításai, és tanítása körül. Elképzelhető az is, hogy egyszerűen több tanulásra van szüksége, hiszen a 39.000 számot felölelő tesztnél nagyobbat nem tudtam futtatni, azon pedig látszódik bizonyos mértékű igazodás, valamint látszik az is, hogy a lineáris tesztadatokra is képes választ közelíteni, azonban minden tanítás után, amit a lineáris adathalmazon végzünk, egyenlőek a predikciók. (a következő értéknél kicsivel kisebb minden bemenetre)