Aláírás-hitelesítés

BME bsc önálló laboratórium

Megvalósítás alapját képező cikk: *LAI, Songxuan; JIN, Lianwen; YANG, Weixin. Online signature verification using recurrent neural network and length-normalized path signature descriptor. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR). IEEE, 2017. p. 400-405.*

A program főbb állítható paraméterei a program elején vannak kilistázva:

```
# window size
ws = 20
# path signature szintje
signiture_level = 5
# tesztadatok mérete százalékban
TESTPERCENTAGE = 10
# db epoch ameddig tanuljon a neurális háló
epochs = 400
# az rnn neuronok száma 1 layerben
hidden_size = 128
# rnn layerek száma
num_layers = 3
# hány iterációnként számoljuk a veszteséget
batch_size = 10
```

Adatbeolvasás:
Az adatokat az SVC2004 adatbázisból olvastam be. A beolvasást és az adatok tározását/rendezését a numpy segítségével csináltam.

Adatfeldolgozás:
Először minden aláíráson végig kell menni és fel kell darabolni azonos hosszúságú részekre. (ablakokba) Minden ablakon ezután el kell végezni egy “path signature” generálást. Ezt az iisigniture könyvtárral végeztem el.

A könyvtár használatáról itt lehet többet olvasni:

- phd-docs/iisignature.pdf at master · bottler/phd-docs · GitHub

A path signiture megértéséhez pedig az alábbi oldalakat javaslom:

- phd-docs/2016_cpeu4_may_neuralnets_chinese.pdf at master · bottler/phd-docs · GitHub (Az elején az RNN-hez is ad egy kis magyarázatot.)
- arxiv.org/pdf/1603.03788.pdf (A matek része)
- Understand rough path iterated integral and how to compute it numerically? - MathOverflow (Itt van példa)

Készítette: Csikós Patrk
Konzulens: Szücs Cintia Lia

Normalizálásra az SKLearn StandardScaler-je ad kényelmes megoldást, amit a következő képpen használok:

```
ss = StandardScaler()
ss.fit(f)
genuine_normalized.append(ss.transform(f))
```

Erre a neurális háló miatt van szükség, mert az csak normalizált adatokra működik helyesen.

Az adatokat ezután meg kell keverni, hogy a neurális háló véletlenül se tanuljon rá az adatok sorrendjére. Itt fontos, hogy az adatokat együtt keverjük az adatok cimkéjével. (Melyik adat, melyik aláírótól származik, illetve hamis vagy eredeti)
Erre az SKLearn shuffle függvénye egy egyszerűen használható megoldást ad.

Ezt követően csak leválasztottam megfelelő számú adatot tesztelés céljára.

Neurális Háló:
A neurális hálóhoz a PyTorch platformot használtam, mert a Tensorflow-ban nehézkes lett volna a Triplet loss megvalósítása veszteségfüggvénynek.

A neurális egy rekurrens neurális háló, aminek a lényege, hogy az előző iteráció kimenetét odaadjuk a következő iterációnak a bemenete mellet.
Erre azért van szükség, mert nem tudjuk előre az adott aláírás méretét. (Illetve aláírásonként eltér a méret.)

- A Pytorch rnn azonban nem képes eltérő darabszámú “idő léptéket” kezelni, ezért ki kell tölteni 0-al.: LSTM - Sequences with different num of time steps · Issue #85 · keras-team/keras (github.com)

A háló minden iteráció után 64 db értékűet ad kimenetnek. Erre a 64 értékre szeretnénk azt elérni, hogy amennyiben hiteles aláírást kapunk az euklideszi távolság a lehető legkisebb, hamis aláírás esetén a lehető legnagyobb legyen. Erre ad veszteségfüggvényt a triplet loss.
Ennél a veszteségfüggvénynél a vizsgált aláíráshoz képest egy helyes aláírás távolságából kivonnyuk egy hamis aláírás távolságát. (Amennyiben ez a szám negatív 0 értéket veszünk veszteségnek.)

Hasznos linkek:

- Triplet Loss — Advanced Intro. What are the advantages of Triplet Loss… | by Yusuf Sarıgöz | Towards Data Science (Triplet loss magyarázat)
- Gated Recurrent Unit (GRU) With PyTorch (floydhub.com) (GRU példa pytorch-ban)
- Triplet Loss with PyTorch | Kaggle (Triplet loss példa PyTorchban (de nem RNN))

Készítette: Csikós Patrk
Konzulens: Szücs Cintia Lia

Döntés menete:
A kiértékelést a hivatkozott cikkben lévő módon valósítottam meg.

$$score^{i}_{test} = \frac{N-1}{2} \sum_{n=1}^{N} d^{i}_{n,test} / \sum_{n=1}^{N} \sum_{m>n} d^{i}_{n,m}.$$

Ahol az i jelöli az alárót, és a $d_{n,m}$ az I ember n,m aláírása közti euklideszi távolságot jelenti. N darab aláírás van az adott embertől.

Kiértékelés:
Az alábbi ábrán látható az általam kapott legjobb 3 eredmény, és azok konfigurációja:

```
# Ablak mérete
ws = 30
# Path signiture szintje
signiture_level = 5
# A teszt adatok százaléka
TESTPERCENTAGE = 10
# Tanulási százalék
epochs = 400

# GRU receptorok száma
hidden_size = 128
# GRU rétegek száma
num_layers = 3
# Bath size
# (Mikor számoljunk gradienst)
batch_size = 10
```

- FAR: 6%
- FRR: 7%
- AER: 7%
- Tanítási idő (rtx 2060): ~20p

```
# Ablak mérete
ws = 20
# Path signiture szintje
signiture_level = 5
# A teszt adatok százaléka
TESTPERCENTAGE = 10
# Tanulási százalék
epochs = 400

# GRU receptorok száma
hidden_size = 128
# GRU rétegek száma
num_layers = 3
# Bath size
# (Mikor számoljunk gradienst)
batch_size = 10
```

- FAR: 9%
- FRR: 4%
- AER: 7%
- Tanítási idő (rtx 2060): ~20p

```
# Ablak mérete
ws = 20
# Path signiture szintje
signiture_level = 4
# A teszt adatok százaléka
TESTPERCENTAGE = 10
# Tanulási százalék
epochs = 400

# GRU receptorok száma
hidden_size = 128
# GRU rétegek száma
num_layers = 3
# Bath size
# (Mikor számoljunk gradienst)
batch_size = 10
```

- FAR: 8%
- FRR: 10%
- AER: 9%
- Tanítási idő (rtx 2060): ~20p

Készítette: Csikós Patrk
Konzulens: Szücs Cintia Lia