Paskaita 1

Dirbtiniai neuroniniai tinklai

Tęsiame analizę (susipažinimą) su metodais ir technologijomis, kurios padeda spręsti labai sparčiai virtualios realybės modeliavimo ir didžiųjų duomenų analizės uždavinius.

Faktiškai nystame tas technologijas, su kuriomis susipažinome Mašininio mokymosi kurse. Galime pasakyti, kad DNT priartina mus prie Dirbtinio intelekto metodų ir galimybių. Svarbesnė prasmė, mes tai vadiname Gilaus mokymosi metodais.

Vėl mus domins prognozės uždavinių arba <u>atpažinimo uždavinių</u> (atpažinti galėt fakto veidus, sandorių tipus, technologinius procesus, rezultatus).

Tipinė apmokymo situacija (kaip ir MM technologijoje):

Turime: a) duomenų aibę (pvz. akcijų kainos pirmadieniais, kai prasideda naujų savaitė vykti biržoj).
b) akcijų kainos antradienį.

Tada vyksta apmokymo procesas su mokytoju (čia ir naudosime naujo technologiją DNT).

Po apmokymo naudosime DNT (jau apmokytą) atsakyti į mus dominančius klausimus.

Nagrinėkime pagrindinių DNT dalių savybes:

1. Apmokymas

Pateikiame duomenų komplektą (pakankamą priimti sprendimą – pvz. nuotraukų ir reikia atpažinti ar joje pavaizduota „katė". Jeigu paduodame labai nedaug duomenų – pvz. 1 ar 2 pikselius, tai tokios informacijos tikrai neužteks net ir žmogui padaryti argumentuotą išvadą, ar šie duomenys atvaizduoja „katę").

Taigi įėjime turime visą porciją duomenų

Išėjime (rezultatas) – vienas taškas (rezultatas)

Parenkame svorių vektorių W, kuris apibrėžia, koks bus DNT atsakymas

Svorių parinkimas – apmokymo tikslas

Aprašykime paprasčiausią DNT

a) input (duomenys įėjime, pvz. vienas skaičius)

b) weight (svoris (nežinotas kol kas)) jį parinksime apmokymo metu)

Mūsų tinklas atlieka prognozę

```
weight = 0.23
def neural_network(input, weight):
    prediction = input * weight
    return prediction
```

Sakykime reikia atlikti prognozę, kai input = 7.2

```
pred = neural_network(input, weight)
print(pred)
```

Ar prognozė yra visada tikli?

Aišku kad ne!

Tokiais atvejais, gavę neteisingą rezultatą, mes vėl grįžtame prie apmokymo proceso (ypač, jei neteisingi rezultatai gaunami dažnai).

Svarstome patikslinti svorio parametro parametrus

Jeigu ir tai nepadeda?

Tai dar ne pasaulio pabaiga!

Sudarome sudėtingesnį DTN, prie kurio daugiau svorio parametrų

$$w_0, w_1, \dots, w_n, \quad n \geq 1.$$

Prie input duomenys yra sudaryti iš reikšmių d_0, d_1, d_2. $\left(I = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix} \right)$

input[0] = d_1, input[1] = d_2, input[2] = d_2.

$n \times 1$ dimensija

Tada apmokymo metu randame optimalius svorius (vektorius eilutė) (1×n dim)

weight[j], j=0,1,2,...,n

```
def neural_network(input, weights):
    prediction = w_sum(input, weights)
    return prediction
```

Prieš tai apibrėžiame pagalbinę f-ją (ji bus naudinga ir kitose situacijose)

```
def w_sum(a, b):
    assert (len(a) == len(b))  # jeigu neišpildyta gausim pranešimą apie klaidą
    output = 0
    for i in range(len(a)):  # 0,1,...,len(a)-1
        output += a[i] * b[i]
    ~~retur~~
    return output
```

Atlikti per realizaciją

Matematinis išreiškimas

wSum (input, weights)
funkcija skaičiuoja dviejų ~~masyvų~~
vektorių input ir weights skaliarinę
sandaugą, t.y. dėl jų galima taip
ir skaičiuoti neural_network realizacijoje:

```
import numpy as np
weights = np.array([0.1, 0.2, 0.05])
def neural_network(input, weights):
    pred = input.dot(weights)
    return pred
```

Taigi, mums nebereikia realizuoti
wSum funkcijos.

Ankstesniame pavyzdyje mes nagrinėjome situaciją, kai duomenų (input) aibę sudarė keli atskiri duomenų vienetai. Tada kiekvienas iš jų įnešė tam tikrą poveikį priimant sprendimą (skaliarinė sandauga su svorių weights vektoriumi) (aišku, jeigu atitinkamas svoris nebuvo lygus 0).

Dabar aptarsime ir kitą galimybę: kai patys duomenys gali būti naudojami ir kito rezultato (sprendimo, vertinimo) gavimui.

Kas tada keičiasi? Nesunku pamatyti, kad tada turime iš naujo apmokyti tinklą ir sudarome kitą svorių weights rinkinį.

Apibendrinkime šį pavyzdį, naudodami algebros technologiją (metodus)

- turime pradinių duomenų vektorių - stulpelį $n \times 1$ dimensijos $I = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \end{pmatrix}$
- turime svorių matricą $m \times n$ dimensijų $W = (w_{ij})$
- rezultatas (prognozė, vertinimas, rodiklis) yra $m \times 1$ dimensijos vektorius-stulpelis

$R = WI \qquad r_i = \sum_{j=0}^{n-1} w_{ij} d_j$

DNT terminologija: turime neuronų paslėptą sluoksnį svorių

Pirmiausia gausime iš paslėptų (tarpinių) įverčių :

I yra duomenų vektorius stulpelis $n \times 1$ dimensijos

W_1 yra svorių matrica $m \times n$ dimens

$R_1 = W_1 I$ yra tarpinis rezult stulpelis $m \times 1$ dimensijos

W_2 yra antroji svorių matrica

$R_2 = W_2 R_1$

Kaip parodysime tolimesnėje analizėje kelių sluoksnių DNT gali daug efektyviau spręsti kai kurios uždavinius (pvz. vaizdų atpažinime).