Paskaita 2

DNT veikimo mechanizmas (matematika, kuri generuoja stebuklus)

Jau Mašininio mokymo kurse mes apibrėžėme technikas, leidžiančias įvertinti sudarytų regresijos modelių tikslumą. Šis aspektas įtrauktas ir į Matematinio modeliavimo paskaitų blokų rinkinį – tik palyginus su realiais fiziniais eksperimentais leidžia įvertinti tikrąjį modelio tikslumą (ir jo naudingumą). Natūralu, kad DNT technologijoje turime skirti svarbų (svarbiausią?) dėmesį prognozės (sprendimų) paklaidų vertinimui

O tada natūraliai gauname matematinių uždavinių:

- kaip parinkti tinklo parametrus (svorius), kad modelio prognozės paklaidos būtų mažiausios (arba bent mažos tiek, kad uždavinys yra praktiškesnis!)
- išmokti parinkti optimalią DNT architektūrą, geriausiai tinkančią duotajai uždavinių klasei

Geroji žinia – jau turime pakankamai efektyvių priemonių (matematinių technikų), kad sudarytume gerus (o kai kada ir netikėtai labai gerus DNT.

Paklaidos skaičiavimas

Šiam klausimui jau esame paskyrę daug dėmesio įvairiose studijų dalyse:

- aproksimacijos paklaida
- statistinio įverčio tikslumas
- mašininio mokymosi prognozės paklaida.

Yra įvairios paklaidos skaičiavimo technikos (normos, funkcijos, ...)

Imkime vektorių $X=(x_1, x_2, \dots, x_M)$ ir jo aproksimaciją/prognozę $Y=(y_1, \dots, y_M)$.

Tada paklaidą galime įvertinti įvairiais būdais

1. Max paklaida: $E_\infty = \max_j |x_j - y_j|$.

(L_∞ normą)

2. Vidutinė paklaida:

$$E_V = \frac{1}{m}\sum_{j=1}^{m} |x_j - y_j| \quad (L_1\text{-norma})$$

3. Kvadratinė paklaida: (jos kvadratas)

$$E^2 = \frac{1}{m}\sum_{j=1}^{m} (x_j - y_j)^2 \quad \left(L_2\text{-norma} = \sqrt{E^2}\right)$$

Taip pat galime nagrinėti šių įverčių santykines versijas

Pvz.:

$$\tilde{E}_\infty = \max_j \frac{|x_j - y_j|}{\varepsilon + |x_j|}$$

Aptarkite skirtingų paklaidų vertinimų (normų) savybes – į kokius procesų bruožus jos atkreipia didžiausią dėmesį)

Dabar apgalvokime, koks bus
tolimesnis mūsų analizės etapas,
kai jau susiskaičiavome paklaidą
(prognozės, sprendimo...)

Aišku, jei paklaida yra maža
(kokia paklaida yra maža?)
tai galime pasinaudoti gauta
prognoze, rekomendacija, atpažinimu.

Bet jeigu paklaida yra didelė?
(kaip tai žinoti apriori, o ne tada,
kai jau dalyvaujame veikme?)

• Taigi abu aspektus reikalinga
pereiti prie mokymosi etapo –
turime išmokyti DNT teikti vertingas
rekomendacijas.

Mokymosi etapas

Šiame etape, vyksta DNT parametrų parinkimas, leidžiantis sumažinti prognozės rezultatų paklaidą.

Matematinis apibrėžimas

Parenkame parametrus taip, kad tinklo funkcija/funkcionalas, apibrėžiantis prognozės rezultatų paklaidą būtų minimali.

a) Sprendžiame optimizavimo uždavinį.

b) Naudojame gradientinio nusileidimo metodą (ir jo modifikacijas).

$$\min_{w_1, w_2, \dots, w_N} f(w_1, w_2, \dots, w_N)$$

$f(w_1, \ldots, w_N)$:

set weights

pred = input * weights

error = $(\text{pred} - \text{goal})^2$

Dažniausiai tenka įvertinti kelių atvejų prognozę.

$$\text{error} = \sum_{j=1}^{M} (\text{pred}_j - \text{goal}_j)^2$$

$$\boxed{\text{pred}_j = \text{input}_j * \text{weights}}$$

Kaip surasti optimalius svorius $w_1, \ldots, w_N$?

Vienas iš elementariu logika paremtų būdu - atlikti paiešką įvairiomis kryptimis

Pvz. imsime w_j ir šio parametro reikšmę pakeisime

$$w_j^{new} = w_j \pm \Delta w.$$

Tada tikriname ar po tokio pakeitimo prognozės paklaida sumažėjo (jei sumažėjo abiem atvejais, tai kuris iš atvejų leido sumažinti paklaidą labiausiai) tarkime, kad tai

$$w_j^{new} = w_j + \Delta w.$$

Tada bandome šį procesą pratęsti

$$\tilde{w}_j^{new} = w_j^{new} \pm \Delta w$$

ir taip gal būt atliekame kelis kartus.

Kaip matysime tai ir yra gradientinio nusileidimo metodo idėja.

```
weight = 0.5
input = 0.5
goal = 0.8
step W = 0.001
                                          (1101
for iter in range(1101):                  -[illegible])
    prediction = input * weight
    error = (prediction - goal) ** 2
    up-predict = input * (weight + step W)
    up-error = (goal - up-predict) ** 2
    down-predict = input * (weight - step W)
    down-error = (goal - down-predict) ** 2
    if (down-error < up-error):
        weight = weight - step W
    if (down-error > up-error):
        weight = weight + step W
```

Klausimai:

a) ar toks algoritmas konverguoja į teisingą atsakymą (visada?).

b) jei ne – tai, kodėl? Ką reikia keisti?

Pagalvokite apie tai;

c) pastovus svorių keitimo žingsnis

d) kaip jį pritaikinti prie uždavinio specifikos?

Rekomendacija (pagalba)

a) minimizuojame paklaidą!

b) prisiminkite gradientinio nusileidimo metodą.

Kol kas nagrinėkime vieno kintamojo

f-ja $f(x)$

Norime rasti $\min f(x) = f(\bar{x}_0)$

$$\arg\min f(x) = \bar{x}_0$$

Iš analizės kurso žinome, kad f-jos

gradientas (vieno kintamojo) 0 atveju

$\frac{df}{dx}$ - išvestinės reikšmė taške $\tilde{x}_0$)

apibrėžia f-jos greičiausio didėjimo

kryptį: Einame antigradiento kryptimi

$$x_{n+1} = x_n - \gamma \frac{df}{dx}(x_n)$$

γ - parenkame tokį, kad $f(x_{n+1}) < f(x_n)$

-12-

Kelių kintamųjų atveju:

$$f = f(x_1, x_2, \ldots, x_m)$$

Gradientas yra vektorius:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_m}\right)$$

$$X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$$

$$\boxed{X^{n+1} = X^n - \gamma \nabla f(X^n)}$$

Kartojam N kartų.

$$error(weight) = (input \cdot weight - goal)**2$$

$$\frac{\partial error(weight)}{\partial weight} = 2 \cdot (input \cdot weight - goal) \cdot input$$

$$\boxed{weight^{n+1} = weight^n - \gamma 2 (input \cdot weight - goal) \cdot input}$$

Mokymo greičiu pvz. $\gamma = 1/2$.