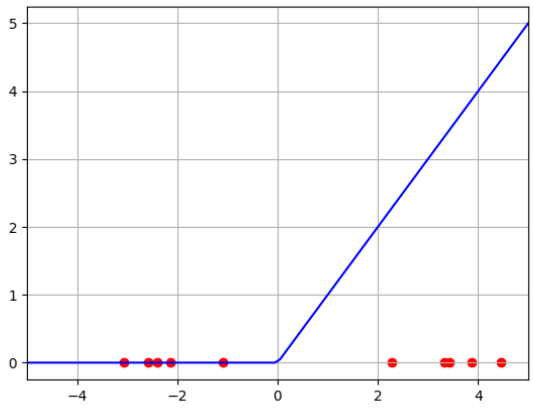
Փաթույթային նորմավորում

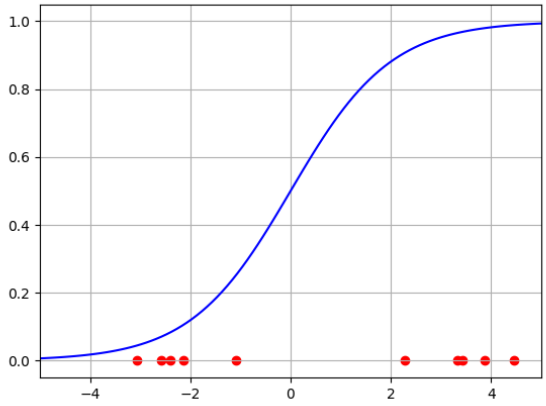
Հայկ Կարապետյան

1 Անհրաժեշտություն

Արդեն գիտենք, որ հնարավոր է նորմավորել մուտքային տվյալները, նորմալիզացիայի եղանակներից ինչ որ մեկը կիրառելով: Նորմավորումը կատարվում էր նրա համար, որպեսզի մակարդակի գծերը լինեն ավելի կլոր և մինիմումի կետ հասնենք ավելի արագ և սահուն: Մեջտեղի շերտերը նույնպես կարող ենք նորմավորել, հիմա ծանոթանանք դրա անհրաժեշտությանը: Պատկերացնենք ունենք շերտ որից տվյալների մի մասը դուրս է գալիս բացասական, իսկ մյուս մասը դրական: Այդ շերտի վրա relu ակտիվացիոն ֆունկցիան կիրառելիս մի մասի արժեքը դառնում է 0, մյուս մասինը x (Նկար 1), իսկ sigmoid ակտիվացիոն ֆունկցիան կիրառելու դեպքում մի մասը շատ մոտ է 1-ին, իսկ մյուս մասը շատ մոտ է 0-ին (Նկար 2):



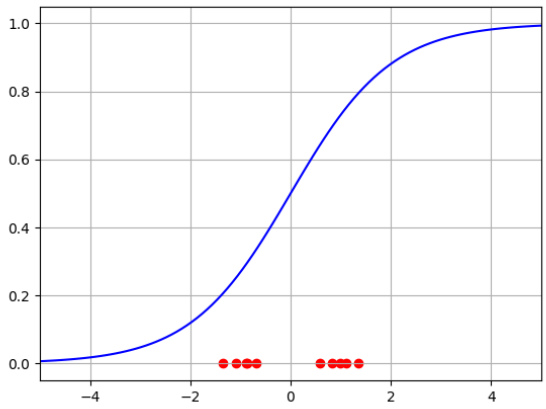
Նկար 1: Արժեքների մի մասի ածանցյալը հավասար է 0-ի



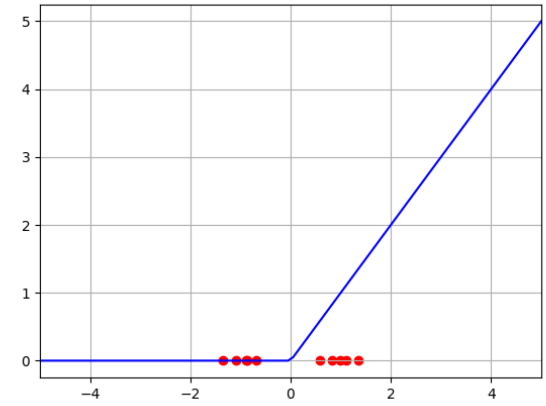
Նկար 2: Այն արժեքները որոնք մոտ են 1-ի կամ 0-ի, ածանցյալը մոտ է 0-ի

Ածանցյալը հաշվելիս relu-ի դեպքում կունենանք 0, երբ բացասական է և 1 երբ դրական է: Sigmoid-ի դեպքում, երբ տվյալները դրական են և նրանց արժեքը մոտ է 1-ի, ածանցյալը մոտ կլինի 0-ի և երբ տվյալները բացասական են և նրանց արժեքը մոտ է 0-ի, ածանցյալը նորից մոտ կլինի 0-ի: Արդեն գիտենք, որ կշիռների թարմացման համար օգտագործում էինք

ածանցյալը և եթե ածանցյալը մոտ է 0-ի, ապա կշիռները գրեթե չեն թարմացվի և ուսուցում տեղի չի ունենա: Ածանցյալների 0-ին մոտ լինելու խնդիրը հայտնի է կորչող գրադիենտ (vanishing gradient) անվանումով: Հետևյալ խնդրին չբախվելու համար մենք կարող ենք միջինացնել մեր տվյալները 0-ի շրջակայքում: sigmoid-ի դեպքում խնդիր գրեթե չի առաջանա (Նկար 3), իսկ relu-ի դեպքում որոշ չափով խնդիրը մնում է (Նկար 4):

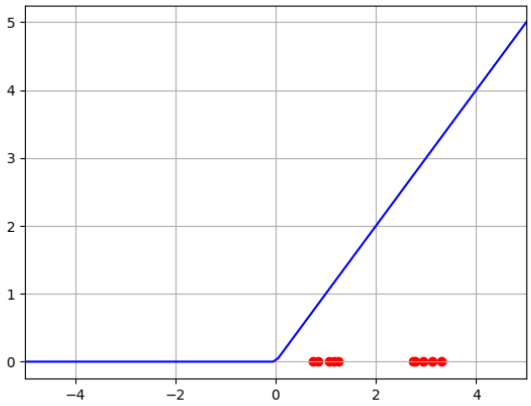


Նկար 3: 0-ին կամ 1-ին մոտիկ արժեքներ չկան



Նկար 4: Խնդիրը չլուծվեց, տվյալների միջինը դարձավ 0,
բայց անհրաժեշտություն կա տեղաշարժելու

Այդ պատճառով ցանցին հնարավորություն ենք տալիս տեղափոխել միջինացված տվյալները, այնպիսի վայր, որտեղ նրանց ածանցյա-լը մոտ չի լինի 0-ի: Այսինքն ցանցը ինքն է որոշում տվյալների միջինը ինչպես փոփոխի մոտեցնի 0-ին, մեծացնի, թե դարձնի բացասական (Նկար 5):



Նկար 5: Ցանցը տեղափոխել է տվյալների միջինը 0-ից դեպի 2

2 Կիրառման եղանակ

Վերը նկարագրված մեթոդը կոչվում է փաթույթային նորմավորում (batch normalization) և այն կիրառվում է ակտիվացիոն ֆունկցիայից առաջ: Նորմավորում կատարելիս գիտենք, որ մակարդակի գծերը ավելի կլորանում են այդ պատճառով ուսուցման արագությունը (learning rate) կարող ենք մեծացնել և ուսուցումը տեղի կունենա ավելի արագ (ավելի շուտ կհասնենք մինիմումի կետին): Փաթույթային նորմավորումը կատա-րվում է փնջերի (batch[1]) վրա հետևյալ կերպ: m-ով նշանակենք փնջի անդամների քանակը: Փունջը նշանակենք $batch$-ով: $batch = \{x_1, \dots, x_m\}$: Պարամետրերը, որը ցանցը պետք է սովորի ուսուցման ընթացքում՝ $\beta,\ \gamma$:

Սկզբից մեզ անհրաժեշտ է հաշվել մեր փնջի միջին արժեքը:

$$\mu_{batch} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

Հետո հաշվենք փնջի վարիացիան (բաշխումը)

$$\sigma^2_{batch} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{batch})^2$$

Նորմավորած տվյալները նշանակենք x գլխարկով:

$$\hat{x_i} \leftarrow \frac{x_i - \mu_{batch}}{\sqrt{\sigma^2_{batch} + \varepsilon}}$$

Էպսիլոնը գումարում ենք հայտարարում 0 չստանալու համար

Վերը կատարված քայլերից հետո մեր տվյալները նորմավորված են (միջինը = 0, բաշխումը = 1) և միայն մնում է նրանց տեղաշարժել այնտեղ, որտեղ մեզ հարկավոր է: Փաթույթային նորմավորումից ստացված տվյալները նշանակենք y_i-ով:

$$y_i \leftarrow \gamma\hat{x_i} + \beta = BN_{\gamma,\beta}(x_i)$$

Նկատենք որ փաթույթային նորմավորումը կատարվում է ակտիվացիոն ֆունկցիայից առաջ:

---

1. Փունջ (batch) - Մուտքային տվյալների համախումբ: Օրինակ՝ ունենք 10 հատ տան մակերես և նրանց համապատասխան գները: Ցանցին մուտքում կարող ենք տալ բոլոր մակերեսները և ելքում պահանջենք բոլոր տան գները միաժամանակ: Սա չի նշանակում, որ ցանցի ելքային շերտը փնջի քանակից կախված փոփոխվում է, այն միշտ մնում է նույնը՝ մեր դեպքում 1 (գին), միայն թե բոլորի համար զուգահեռ վերադարձնում է այդ մի ելքը:

3 Փոխարինենք նորմավորումը նեյրոնով

Ինչպես գիտենք տվյալների նորմավորում կատարելիս անում ենք հետևյալ գործողությունը.

$$x_i = \frac{x_i - \mu}{\sigma} = \frac{1}{\sigma}x_i - \mu \tag{1}$$

$\frac{1}{\sigma}$ – և նշանակենք w-ով, իսկ μ – և՝ b-ով

$$x_i = wx_i + b$$

Ստացանք, որ նորմավորումը համարժեք է տվյալները մեկ նեյրոնի միջով անցկացնելու

Այս ամենը ճիշտ կլիներ, եթե մենք մեր ցանցին ամեն քայլի ժամանակ մուտքում տայինք բոլոր առկա տվյալները այլ ոչ թե փունջ առ փունջ։ Փունջ առ փունջ տալիս, նեյրոնային ցանցը չի կարող սովորել μ և σ-ի արժեքները, քանի որ դրանք ստացվում են ամբողջ առկա տվյալների միջոցով.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Ստացանք որ տվյալների նորմավորումը նեյրոնային ցանցը չի կարող սովորել։ Դիտարկենք փաթույթային նորմավորումը։ Հնարավոր է արդյոք այն փոխարինել նեյրոնով։ Վերցնենք դեպք, երբ տվյալների միջինը և ստանդարտ բաշխումը չենք փոփոխում (այսինքն μ չենք հավասարեցնում 0-ի և σ չենք հավասարեցնում 1-ի)։ Այդ դեպքում փաթույթային նորմավորման շերտը պետք է սովորի ցանկացած միջինի և բաշխման դեպքում բերի իրեն հարմար միջինի և բաշխման։ Բայց դա գրեթե հնարավոր չէ։ Իսկ երբ ամեն անգամ տվյալների միջինը բերենք 0-ի իսկ բաշխումը 1-ի, փաթույթային նորմավորմանը մնում է սովորել ինչպես տեղաշարժել 0 միջինից և 1 բաշխումից, ոչ թե ամեն անգամ տարբեր արժեքներից։ Ամբողջ տվյալների վրա միանգամից ուսուցում կատարելու ժամանակ տեսականորեն անիմաստ է կատարել նորմավորում ($BN(x) = wx + b$), բայց պրակտիկայում հնարավոր է, որ w-ն և b-ն չկարողանան սովորեն միջինի և բաշխման արժեքները, այդ պատճառով մենք հեշտացնում ենք ցանցի գործը և միանգամից մուտքում տալիս ենք նորմավորված տվյալներ։

4 Առաջացող հարցեր

1. Կարո՞ղ ենք կիրառել ստոխաստիկ գրադիենտային վայրէջք $(batch = 1)$, երբ օգտագործում ենք փաթույթային նորմավորում։ Երբ մուտքային տվյալների քանակը հավասար է 1-ի, տվյալների միջինը հավասար է լինում հենց այդ մեկ տվյալին և ամեն տվյալից երբ հանենք տվյալների միջինը կստանանք 0։ Այդ պատճառով ստոխաստիկ գրադիենտային վայրէջքի դեպքում չենք օգտագործում փաթույթային նորմավորում։

2. Այդ դեպքում ի՞նչ միջին և բաշխում ենք օգտագործելու թեստավորման ժամանակ։ Չէ որ այս դեպքում նույնպես մեր ցանցը հնարավոր է ստանա մի տվյալ (batch=1)։ Մենք օգտագործելու ենք ուսուցման ժամանակ ստացած միջինները և բաշխումները։ Օրինակ՝ 3-րդ շերտի վրա դրված է փաթույթային նորմավորում։ Այն ամեն անգամ նախորդ շերտից ստանում է տարբեր մուտքային տվյալներ, հաշվում է դրանց համար միջինը և բաշխումը ու նորմավորում դրանք։ Մեզ անհրաժեշտ է ինչ որ կերպ բոլոր միջինները միջինացնենք և վերջին տվյալներին ավելի շատ ուշադրություն դարձնենք քան սկզբիններին։ Վերջին միջիններին մեզ պետք է ավելի շատ ուշադրություն դարձնենք, քանի որ շերտերի կշիռները ժամանակի ընթացքում ավելի լավ են սովորում և փաթույթային նորմավորման շերտին մուտքում գալիս են ավելի լավ սովորած արժեքներ և դրանց համար է որոշվում μ-ի և σ-ի արժեքը։ Վերջիններին ավելի շատ ուշադրություն դարձնելու համար կիրառում ենք էքսպոնենցիալ շարժվող միջին (exponential moving average)։

3. Իսկ ի՞նչ է կատարվում այդ շերտի բիասների հետ։ Դիտարկենք հետևյալ օրինակը՝ ունենք մի նեյրոն, տվյալների պարամետրերի քանակը մեկ է (մակերես), $batch\_size = n$։ Նեյրոնից դուրս եկող արժեքները կլինեն՝ $\{wx_1+b, wx_2+b, \dots, wx_n+b\}$։ Այս դեպքում $\mu = \frac{x_1 + x_2 + \cdots + x_n}{n} w + b$։ Կատարելով (1) գործողությունը, կստանանք.

$$x_i = \frac{wx_i + b - \frac{x_1 + x_2 + \cdots + x_n}{n} w + b}{\sigma} = \frac{w\left(x_i - \frac{x_1 + x_2 + \cdots + x_n}{n}\right)}{\sigma}$$

Տեսանք որ փաթույթային նորմավորման ժամանակ բիասները կրճատվում են, այսինքն այդ շերտում կարող ենք բիասներ չօգտագործել (նեյրոնից դուրս եկող արժեքը $= wx$)։