

Գեներատիվ մրցակցող ցանցերի կիրառումը նկարի տեսքով թաքնագրության կրիչ ստեղծելու համար

Նախաբան

Թաքնագրությունը գաղտնի տեղեկատվությունը ոչ գաղտնի տեղեկատվության (կոնտեյներ) մեջ թաքցման մեթոդների հավաքածու է: Իսկ թաքնավերլուծությունը (Steganalysis), մի գործընթաց է, որն ուղղված է պարզելուն, թե արդյո՞ք հաղորդագրությունը պարունակում է թաքնված ինֆորմացիա, և հնարավորության դեպքում այն վերականգնելուն: Թաքնված ինֆորմացիայի ներկայությունը հայտնաբերելու համար սովորաբար օգտագործվում է երկուական դասակարգիչ (Binary classifier): Սույն ուսումնասիրության մեջ ներկայացվելու է մի մոդել, որը ստեղծում է նկար-կոնտեյներներ, հիմնված՝ խորը Պարուրման Ստեղծարար Մրցակցող Ցանցերի (Deep Convolutional Generative Adversarial Networks, կրճատ՝ DCGAN) վրա: Այս մոտեցումը թույլ է տալիս առաջարկել ավելի թաքնակայուն կոնտեյներ, ներդրված հաղորդագրությամբ, օգտագործելով ստանդարտ թաքնագրային ալգորիթմներ:

Այս թեմայի շուրջ 2016թ.-ին կատարվել է հետազոտություն, որի ընթացքում փորձել են գեներացնել մարդկանց դեմքեր: Մոդելը հաջողությամբ մոլորեցրել է թաքնագրային վերլուծիչին, սակայն որոշ դեպքերում մարդու աչքը գեներացված նկարները հեշտությամբ կարող էր տարբերել իրականից, քանզի մոդելին՝ ուսուցման ժամանակ, տրամադրվել էին տարբեր սեռի մարդկանց դեմքեր, սակայն չէին հաշվի առել այդ հանգամանքը:

Ուսուցմանը մասնակցելու են միանգամից 3 մոդել: Դրանք են՝

1. Գեներացնող մոդել (Գեներատոր - Generator) - G
2. Տարբերակող մոդել (Տարբերակիչ - Discriminator) - D
3. Թաքնավերլուծող մոդել (Թաքնավերլուծիչ - Steganalyser) - S

Առաջին մոդելը՝ գեներատորը, պատասխանատու է նկարներ գեներացնելու համար, այն պետք է այնպիսի նկարներ գեներացնի, որ հնարավոր չլինի տարբերել իրական նկարներից: Այս խնդրի լուծման համար օգտագործվելու է երկրորդ մոդելը՝ տարբերակիչը, որի խնդիրն է լինելու տարբերել իրական նկարը կեղծից (կեղծ են բոլոր այն նկարները որոնք ստեղծել է G գեներատորը): Այս ամենից հետո գործի է անցնում 3-րդ մոդելը՝ վերլուծիչը, որի խնդիրն է պարզել արդյո՞ք տրված նկարում առկա է թաքնագրված ինֆորմացիա, թե՞ ոչ: D վերլուծիչին ուսուցման ընթացքում տրամադրվելու են գեներատորի նկարները, որոնք արդեն պարունակում են թաքնագրված ինֆորմացիա, ինչպես նաև սովորական նկարներ, որոնք չեն պարունակում ոչ մի թաքնագրված ինֆորմացիա:

Այսպիսով D տարբերակիչն ու S վերլուծիչը բարելավելու են իրենց արդյունքը՝ հիմնվելով G գեներատորի տրամադրած և սովորական նկարների վրա, իսկ G-ն

բարելավելու է իր արդյունքը՝ հիմնվելով D-ի և S-ի արդյունքի վրա: Հենց այստեղ էլ առաջ է գալիս մրցակցող ցանցերի գաղափարը, քանզի ստացվում է, որ ցանցերը մրցում են միմյանց հետ, թե ում արդյունքն ավելի լավը կլինի:

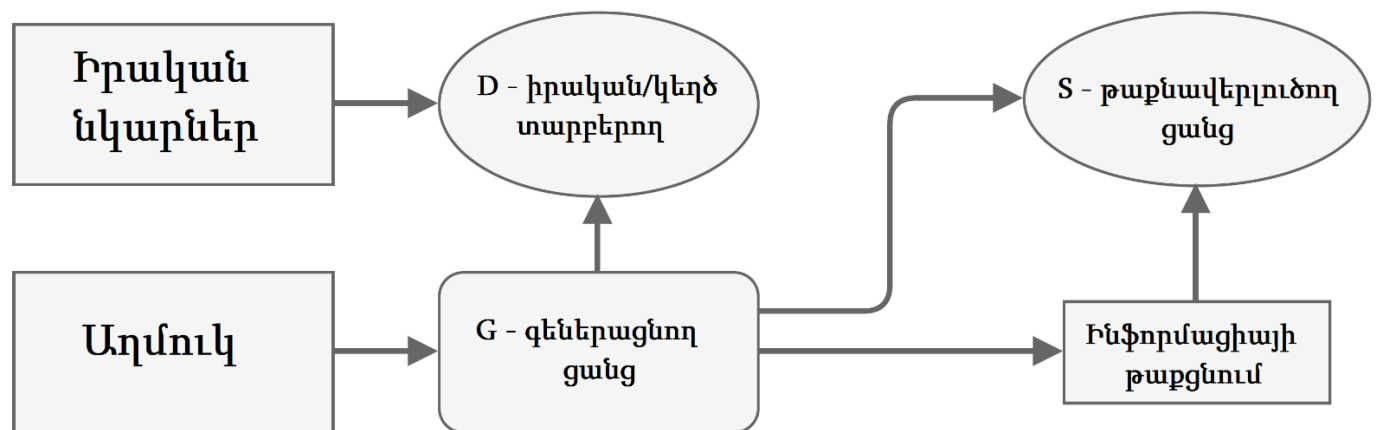
Վերջերս մշակված մրցակցող ցանցերը (GAN, տես Goodfellow(2014)) հզոր գեներացնող մոդելներ են, որոնց հիմնական գաղափարը գեներատորի և տարբերակիչի ուսուցումն է միմյանքս խաղի միջոցով: G մոդելը մուտքին ստանում է պատահական՝ այսպես ասած անիմաստ նկար, որի հիման վրա փորձում է ստեղծել հնարավորինս իրականին մոտ պատկեր, իսկ D-ն ձգտում է տարբերակել իրական պատկերները կեղծերից:

Գոյություն ունեն սմանատիպ ցանցերի տարբեր մոդիֆիկացիաներ՝

- Խորը Պարուրման Ստեղծարար Մրցակցող Ցանցեր (DCGAN, տես՝ Radford (2015))
 - այս մոդելը Ստեղծարար Մրցակցող Ցանցի (GAN) փոփոխություն է, որը մասնագիտացված է պատկերների առաջացման ուղղությամբ
- Պայմանական Մրցակցող Ցանցեր
 - թույլ է տալիս ստեղծել որևէ դասի օբյեկտներ, տես Mirza & Osindero (2014);
- Պատկերների առաջացում՝ հիմնված տեքստային նկարագրության վրա, տես Reed (2016):

Թաքնագրվող գաղտնի ինֆորմացիան, ինչպես նաև կոնտեյները, կարող է ներկայացված լինել տարբեր տեսքով՝ նկարի, տեքստի, տեսահոլովակի, ձայնագրության և այլն: Այս ուսումնասիրության մեջ կատարվելու է տեքստի թաքնագրում նկարում և օգտագործվելու է DCGAN տեսակը:

Ներքևում ներկայացված են մոդելները և նրանց միջև կապերը՝



Մեքենայական ուսուցում (Machine Learning – ML)

Նախքան անցնելը բուն թեմային, ծանոթանանք մեքենայական ուսուցման հետ: Արթուր Սամուելն այն նկարագրում է այսպես. «Մեքենայական ուսուցումը մի տեխնոլոգիա է, որը համակարգիչներին հնարավորություն է տալիս սովորելու, առանց բացահայտ ծրագրավորված լինելու»: Սա, իհարկե, ոչ պաշտոնական ձևակերպում է, սակայն լավ պատկերացում է տալիս:

Մեքենայական ուսուցման ալգորիթմներից են.

- Վերահսկվող ուսուցում (Supervised learning)
- Չվերահսկվող ուսուցում (Unsupervised learning)
- Ուսուցում ամրապնդմամբ (Reinforcement learning)
- Խորհրդատու համակարգ (Recommender system)

Վերահսկվող ուսուցման դեպքում մեքենային տրվում է մուտքային տվյալների հավաքածու և այդ տվյալներին համապատասխան ելքային արժեքները: Այսպիսով այս ուսուցման դեպքում մեքենային հայտնի են ամեն մի մուտքային ինֆորմացիային համապատասխանող ելքային արժեքը կամ արժեքները:

Վերահսկվող ուսուցում (Supervised learning)

Վերահսկվող ուսուցման խնդիրները դասակարգվում են հետևյալ 2 տիպերի.

- Ռեգրեսսիայի խնդիրներ (Regression problems)
- Դասակարգման խնդիրներ (Classification problems)

Ռեգրեսսիայի խնդիրներում փորձում ենք կանխատեսել անընդհատ ֆունկցիայի արժեքներ, ինչը նշանակում է, որ մենք փորձում ենք մուտքային փոփոխականները համապատասխանեցնել ինչ-որ անընդհատ ֆունկցիայի ելքային արժեքներին: Դասակարգման հարցում մենք փոխարենը փորձում ենք կանխատեսել ընդհատ ելքային արժեքներ: Այլ կերպ ասած, մենք փորձում ենք մուտքային փոփոխականները համապատասխանեցնել դիսկրետ կատեգորիաների:

Ռեգրեսսիայի խնդրի օրինակ՝ «Տրված մարդու նկարից որոշել նրա տարիքը»:

Դասակարգման խնդրի օրինակ՝ «Տրված է որևէ հիվանդի ուռուցքի մասին ինֆորմացիա, որոշել արդյո՞ք ուռուցքը չարորակ է, թե՞ բարորակ»:

Չվերահսկվող ուսուցում (Unsupervised learning)

Չվերահսկվող ուսուցումը հնարավորություն է տալիս լուծել այնպիսի խնդիրներ, որոնց ելքային արժեքների մասին կամ քիչ ինֆորմացիա ունենք, կամ ընդհանրապես չգիտենք, թե ինչ տեսքի պետք է լինեն: Մենք կարող ենք ստանալ մի այնպիսի ելքային

տվյալի կառուցվածք, որի վրա մուտքային տվյալի ազդեցությունն անգամ չգիտենք: Այդ կառուցվածքը հնարավոր է ստանալ տվյալները համախմբելու արդյունքում՝ հիմնված մուտքային տվյալի փոփոխականների միջև կապերի վրա:

Չվերահսկվող ուսուցման ժամանակ կանխատեսման արդյունքների վրա հիմնված հետադարձ կապ չկա:

Օրինակներ՝

Կլաստերիզացիա. Վերցնել 1,000,000 տարբեր գեների հավաքածու և ավտոմատացնել այդ գեների խմբավորումն այնպիսի խմբերում, որոնք ինչ-որ կերպ նման են կամ կապված են տարբեր փոփոխականների հետ՝ ինչպիսիք են կյանքի տևողությունը, գտնվելու վայրը, դերը և այլն:

Ոչ կլաստերիզացիա. «Կոկտեյլային երեկույթի ակզորիթմը», թույլ է տալիս գտնել կառուցվածք քառասյին միջավայրում (այսինքն, առանձնացնել մարդու խոսակցության ձայնը երեկույթում հնչող երաժշտությունից):

Որոշ նշանակումներ

Կատարենք մի քանի նշանակումներ, որոնք կոգտագործվեն հետագայում:

Դիցուք ունենք հետևյալ տվյալները՝

| | | | |
|-----------------|-----|-----------------|----------------|
| X_1 | ... | X_n | Y |
| $Input^{(1)}_1$ | ... | $Input^{(1)}_n$ | $Output^{(1)}$ |
| ... | ... | ... | ... |
| $Input^{(m)}_1$ | ... | $Input^{(m)}_n$ | $Output^{(m)}$ |

X_1, X_2, \dots, X_n -ը մուտքային պարամետրերի նշանակումներն են, Y -ը՝ ելքային պարամետրի նշանակումը: $Input^{(i)}_1, Input^{(i)}_2, \dots, Input^{(i)}_n$ -ը մուտքային պարամետրերի արժեքներն են (տվյալի հատկություններ), իսկ $Output^{(i)}$ -ն՝ ելքային պարամետրի արժեքն է, որտեղ՝ $i=1,2,\dots,m$: Հարմարավետության համար $Input^{(i)}_1, Input^{(i)}_2, \dots, Input^{(i)}_n$ -ը նշանակենք $x^{(i)}$ -ով, իսկ $Output^{(i)}$ -ն՝ $y^{(i)}$ -ով: Պարզ է, որ՝ n -ը մուտքային պարամետրերի քանակն է:

$(x^{(i)}, y^{(i)})$ զույգն անվանում ենք ուսուցման օրինակ (training example), իսկ դրանց ցուցակը՝ ուսուցման տվյալներ (training set): Այսինքն m -ը՝ ուսուցման տվյալների քանակն է:

Այժմ կարող ենք տալ վերահսկվող ուսուցման ավելի ֆորմալ ձևակերպում՝ «Վերահսկվող ուսուցման նպատակն է՝ տրված ուսուցման տվյալների հիման վրա ձևավորել մի այնպիսի $h : X \rightarrow Y$ ֆունկցիա, որ $h(x)$ -ի ելքային արժեքը բավականին մոտ լինի համապատասխան y -ի արժեքին»: h ֆունկցիան անվանում են «հիպոթեզ»:

Ինչքան $h(x)$ -ի արժեքը մոտ լինի համապատասխան y -ի արժեքին, այնքան ավելի ճիշտ արդյունքներ կտա մեր մեքենայական ուսուցման մոդելը:

Բնականաբար $h(x)$ -ը ունի գործակիցներ, նշանակենք այդ գործակիցները $\theta_0, \theta_1, \dots, \theta_n$ -ով, այս պատճառով $h(x)$ -ը որոշ դեպքերում կնշանակենք $h_{\theta}(x)$:

Հասկանալի է, որ մեր խնդիրը հենց այդ θ -ների արժեքները գտնելու մեջ է կայանում, քանզի հետագայում երբ արդեն մեր մոդելը բավարար չափով ուսուցանված կլինի, և ունակ կլինի գուշակել ճիշտ արժեքներ, նրան տրվելու են X_1, X_2, \dots, X_n արժեքները և քանզի այն ունի արդեն հաշվարկած $\theta_0, \theta_1, \dots, \theta_n$ արժեքները, ընդամենը պետք է հաշվի $h_{\theta}(x)$ -ի արժեքը:

Արժեքի ֆունկցիա (Cost Function)

$h(x)$ -ի արժեքների ճշտությունը կարելի է գնահատել **արժեքի ֆունկցիայի** միջոցով: Այն իրենից ներկայացնում է $h(x)$ -ի բոլոր ելքային արժեքների և իրական y -ների արժեքների միջինացված տարբերություն:

Ահա բանաձևը՝

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h(x_i) - y_i)^2$$

Ավելի պարզ այն կարող ենք գրել հետևյալ կերպ՝ $\frac{1}{2} \bar{x}$, որտեղ \bar{x} -ը $(h_{\theta}(x_i) - y_i)$ -ի քառակուսային միջինն է, այսինքն՝ գուշակված և իրական արժեքի տարբերությունը:

Այս ֆունկցիան նաև կոչվում է «Քառակուսային սխալի ֆունկցիա» (“Squared error function”): Քառակուսային միջինը բաժանվել է 2-ի՝ հետագա հաշվարկների հարմարավետության համար, քանի որ դրա միջոցով $(h_{\theta}(x_i) - y_i)^2$ -ի ածանցումից ստացված 2 բազմապատիկը կվերանա:

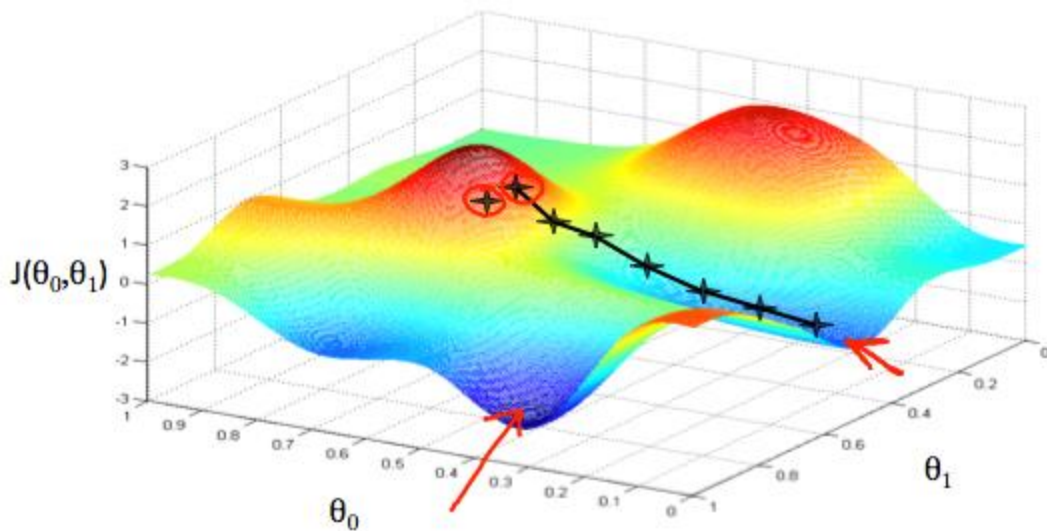
Ստացվեց, որ մեր խնդիրը կայանում է $J(\theta_0, \theta_1, \dots, \theta_n)$ -ը մինիմիզացնելու մեջ, որն ավելի ֆորմալ կարող ենք ներկայացնել հետևյալ կերպ՝

$$\underset{\theta_0, \theta_1, \dots, \theta_n}{\text{minimize}} J(\theta_0, \theta_1, \dots, \theta_n)$$

Նվազող գրադիենտ (Gradient Descent)

Այսիպսով արդեն պարզաբանվեց, թե ինչ է հիպոթեզ ֆունկցիան և թե ինչպես կարելի է չափել նրա ճշտությունը: Այժմ անհրաժեշտ է որոշել հիպոթեզի պարամետրերը:

Դիտարկենք հիպոթեզ ֆունկցիայի պարզեցված օրինակ, որն ունի ընդհամենը 2 պարամետր՝ θ_0 և θ_1 : Պատկերենք այդպիսի հիպոթեզի արժեքի ֆունկցիայի օրինակ:



Այստեղ պետք է հստակ պատկերացնել, որ մենք չենք գծում հիպոթեզի գրաֆիկը, այլ փոխարենը գծում ենք նրա **արժեքի ֆունկցիայի** գրաֆիկը, որը ցույց է տալիս, թե θ_0 -ի և θ_1 -ի արժեքների համար ինչքանով է հիպոթեզը շեղված սպասվելիք արժեքներից: Հասկանալի է, որ պետք է գտնել տվյալ գրաֆիկի վրայի ամենացածր կետը, որի θ_0 և θ_1 արժեքներն էլ հենց կլինեն մեր հիպոթեզի որոնելի պարամետրերի արժեքները (վերևի նկարում կարմիր սլաքներով նշված են տվյալ գրաֆիկի մինիմումները): Զանգի արժեքի ֆունկցիան հիմնականում իրենից ներկայացնում է բարդ մաթեմատիկական բանաձև, այն դժվար է գծել, կամ գտնել, թե θ -ի որ արժեքների դեպքում է այն ընդունում մինիմալ արժեք: Հենց այս խնդիրը լուծելու համար օգտագործվում է նվազող գրադիենտը:

Նշվածն իրականացնելու համար կօգտագործենք արժեքի ֆունկցիայի ածանցյալը: Ածանցյալը ցույց է տալիս տվյալ կետում շոշափողի ուղղությունը, ինչն էլ ինֆորմացիա է տալիս այն մասին, թե որ ուղղությամբ պետք է շարժվել: Ամեն քայլին շարժվում ենք այն ուղղությամբ, որն ամենաշատն է նվազեցնում արժեքի ֆունկցիան:

Յուրաքանչյուր քայլի չափը որոշվում է α պարամետրի միջոցով, որը կոչվում է ուսուցման գործակից (learning rate): Օրինակ, վերը նշված գրաֆիկում յուրաքանչյուր «աստղի» հեռավորությունը ներկայացնում է քայլի հեռավորությունը՝ պայմանավորված α պարամետրով: Փոքր α -ն համապատասխանում է փոքր քայլի, իսկ մեծը՝ մեծ քայլի: Զայլի ուղղությունը, որոշվում է $J(\theta_0, \theta_1)$ -ի մասնակի ածանցյալով: Կախված այն բանից, թե որտեղից ենք սկսում դիտարկել գրաֆիկը, հնարավոր է տարբեր մինիմումների հասնել: Վերևում պատկերված են երկու տարբեր սկզբնակետեր (վերցված են կարմիր շրջանագծերի մեջ), որոնք հասնում են երկու տարբեր մինիմումների:

Ընդհանուր դեպքի համար նվազող գրադիենտի ալգորիթմը կլինի.

կրկնել հեկյալը մինչև զուգամիտում՝ $\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1, \dots, \theta_n)$ որտեղ՝

$j = 0, 1, \dots, n$ ներկայացնում է հատկության հերթական համարը: Այն անվանում են նաև թարմացման օրենք (update rule):

Մեր օրինակի համար՝ $n = 1$:

Յուրաքանչյուր իտերացիային պետք է միաժամանակ թարմացնել բոլոր $\theta_0, \theta_1, \dots, \theta_n$ պարամետրերը: Այսինքն նախ տվյալ իտերացիայի համար հաշվարկել բոլոր θ -ների արժեքները՝ θ' , որից հետո θ -ին վերագրել θ' : Եթե կամայական θ_j -ի արժեքը թարմացնենք նախքան բոլոր θ -ների արժեքները հաշվարկելը, ապա կստանանք սխալ պատասխան:

Պետք է հաշվի առնել, որ կարևոր է α -ի ճիշտ ընտրությունը, քանզի դրանով է պայմանավորված ալգորիթմի զուգամիտման ժամանակը: Եթե ալգորիթմը չի զուգամիտում կամ շատ ժամանակ է պահանջում մինիմումին հասնելու համար ապա α քայլաչափը սխալ է ընտրված:

Այստեղ կարող է հարց առաջանալ, թե արդյո՞ք հնարավոր է հասնել մինիմումի՝ α -ի անփոփոխ արժեքի դեպքում: Պատասխանը պարզ է դառնում, երբ հաշվի ենք առնում այն հանգամանքը, որ, քանզի ամեն քայլ անելուց մենք ավելի ենք իջնում արժեքի ֆունկցիայի կորով ներքև, հետևաբար ամեն քայլի հետ մեկտեղ ածանցյալի արժեքը նվազում է: Իսկ դա նշանակում է, որ անգամ, եթե α -ն հաստատուն պահենք, այնուամենայնիվ $\alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1, \dots, \theta_n)$ արտադրյալը ամեն քայլին կնվազի և հասնելով որևէ մինիմումի այն կհավասարվի 0-ի և հետագա քայլերը ոչ մի կերպով չեն ազդի θ -ների արժեքների վրա:

Հեշտությամբ կարելի է համոզվել, որ, եթե մեր հիպոթեզն ունի գծային տեսք՝

$$h_{\theta}(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_n X_n$$

ապա թարմացման օրենքի մեջ $J(\theta)$ -ի արժեքը տեղադրելուց հետո թարմացման օրենքի տեսքը կլինի՝

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

որտեղ՝ $j := 0 \dots n$:

Այստեղ և հետագայում կընդունենք, որ $x_0^{(i)} = 1$, բոլոր i -երի համար: Սա արվում է բանաձևերը հարմար ներկայացնելու համար: Ստացվեց, որ գծային հիպոթեզն ունի հետևյալ տեսքը՝

$$h_{\theta}(x) = \theta_0 X_0 + \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_n X_n$$

որտեղ, ինչպես նշվեց, $X_0 = 1$:

Հատկության մասշտաբավորում (Feature Scaling)

Մենք կարող ենք արագացնել նվազող գրադիենտի աշխատանքը՝ բերելով բոլոր մուտքային պարամետրերը մոտավորապես նույն տիրույթի թվերի: Դա կապված է այն բանի հետ, որ որ θ -ն ավելի արագ է հասնում մինիմումին փոքր միջակայքերում և ավելի դանդաղ՝ մեծ միջակայքերում, հետևաբար այն տատանվելով է այն տատանվելով է ձգտում մինիմումին, երբ փոփոխականները շատ անհավասար են:

Դա կանխելու համար կարող ենք այնպես փոփոխել հատկությունները (մուտքային պարամետրերը), որ նրանք ընկնեն մոտավորապես միևնույն թվային տիրույթ: Իդեալական դեպքում՝

$$-1 < x_i < 1$$

կամ՝

$$-0.5 < x_i < 0.5$$

Սրանք պարտադիր պահանջներ չեն, մենք ընդամենը փորձում ենք կրճատել հաշվարկների ժամանակը: Նպատակն է՝ բերել բոլոր մուտքային փոփոխականները միևնույն տիրույթի:

Հատկության մասշտաբավորումն ու միջինով նորմալացումը (mean normalization) այն երկու մեթոդներն են, որոնք կօգնեն լուծել այդ խնդիրը: Առաջինը ենթադրում է մուտքային տվյալների բաժանում նրանց մեծագույն և փոքրագույն արժեքների տարբերության վրա: Միջինով նորմալացման դեպքում պետք է մուտքային փոփոխականից հանել մուտքային տվյալների միջին արժեքը, ապա նոր բաժանել մեծագույն և փոքրագույն արժեքների տարբերության վրա: Ստացվեց, որ այս երկու մեթոդների իրականացման համար անհրաժեշտ է փոփոխել մուտքային պարամետրերը՝ համապատասխան ներքևի բանաձևի:

$$x_i := \frac{x_i - \mu_i}{s_i}$$

որտեղ s_i -ն i -րդ հատկության մեծագույն և փոքրագույն արժեքների տարբերությունն է, իսկ μ_i -ն՝ այդ հատկության բոլոր արժեքների միջինը: Նշենք, որ s_i -ին կարող ենք ընդունել հավասար միջին քառակուսային շեղմանը, և այդ դեպքում ստացված արժեքները կտարբերվեն նախորդ տարբերակով ստացված արժեքներից:

Նշվածի օրինակ կարող է ծառայել հետևյալը՝ եթե x_i -ն ներկայացնում է բնակելի տան բարձրություն, և գտնվում է 4-ից 34 միջակայքում, իսկ այդ հատկության բոլոր արժեքների միջինը հավասար է 18-ի, ապա $x_i := \frac{\text{արժեք}-18}{30}$:

Ուսուցման գործակից (Learning Rate)

Նվազող գրադիենտն իրականացնելուց հետո անհրաժեշտ է հետևել ալգորիթմի աշխատանքին (մոդելի ուսուցման պրոցեսին) և հասկանալ արդյո՞ք այն ճիշտ է աշխատում:

Պատկերացում կազմելու համար, թե ինչքան լավ է սովորում մոդելը, անհրաժեշտ է գծել արժեքի ֆունկցիայի՝ $J(\theta)$ -ի, կախումը *իտերացիաների քանակից*: Պարզ է, որ, եթե ամեն ինչ ճիշտ է աշխատում, ապա ամեն իտերացիայից հետո $J(\theta)$ -ի արժեքը պետք է նվազի՝ ձգտելով 0-ի: Հետևաբար, եթե գրաֆիկը աճում է, ապա ինչ որ բան այն չէ: Հիմնականում դրա պատճառը α -ի մեծ արժեքն է լինում. անհաժեշտ է նվազեցնել α -ի արժեքը:

Հարկ է նշել՝ ապացուցված է, որ, եթե ուսուցման գործակից α -ն բավարար չափով փոքր է ընտրված, ապա $J(\theta)$ -ն նվազում է ամեն իտերացիային: Սակայն եթե այն շատ փոքր է ընտրված, ապա $J(\theta)$ -ն կարող է շատ դանդաղ նվազել:

Կարելի է համարել որ մոդելը բավարար չափով ուսուցանվել է այն պահին, երբ $J(\theta)$ -ի փոփոխությունն ինչ-որ իտերացիայից հետո ավելի փոքր է որևէ E արժեքից: E -ն կամայապես ընտրված փոքր թիվ է, օրինակ՝ 10^{-3} : Գործնականում դժվար է ընտրել E -ի օպտիմալ արժեք:

Գրականություն

- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. pp. 2672–2680, 2014
- Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015
- Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 2014
- Scott Reed, Zeynep Akata, Xincheng Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. 2016