**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 մինիմաքս 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 նմանատիպ 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 ծրագրաված լինելու»: Սա, իհարկե, ոչ պաշտոնական ձևակերպում է, սակայն լավ պատկերացում է տալիս։

Մեքենայական ուսուցման ալգորիրթմները երկուսն 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 խնդրներում 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 համապատասխանեցնենք 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 ուսումը հնարավորություն է տալիս լուծել այնպիսի խնդիրներ, որոնց ելքային արժեքների մասին կա՛մ քիչ ինֆորմացիա ունենք, կա՛մ 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 ալգորիթմը», թույլ է տալիս գտնել կառուցվածք քաոսային միջավայրում (այսինքն, առանձնացնել մարդու խոսակցության ձայնը երեկույթում հնչող երաժշտությունից):

**Որոշ նշանակումներ**

Կատարենք մի քանի նշանակումներ, որոնք կոգտագործվեն հետագայում:

Դիցուք ունենք հետևյալ տվյալները՝

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | … | Xn | Y |
| Input(1)1 | … | Input(1)n | Output(1) |
| … | … | … | … |
| Input(m)1 | … | Input(m)n | Output(m) |

*X1, X2, … Xn-ը* մուտքային պարամետրերի նշանակումներն են, *Y-ը՝* ելքային պարամետրի նշանակումը։ *Input(i)1, Input(i)2, … Input(i)n-ը* մուտքային պարամետրերի արժեքներն են, իսկ *Output(i)*-ն՝ ելքային պարամետրի արժեքն է, որտեղ՝ *i=1,2,…,m։* Հարմարավետության համար *Input(i)1, Input(i)2, … 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 → Y* ֆունկցիա, որ *h(x)-ի* ելքային արժեքը բավականին մոտ լինի համապատասխան *y-ի* արժեքին»։ *հ* 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 պարամետրեր, նշանակենք այդ պարամետրերը   
*θ0,θ1, … θn-ով,* այս պատճառով *h(x)-ը* որոշ դեպքերում նշանակում են *hθ(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 *X1, X2, … Xn* արժեքները և քանզի այն ունի արդեն հաշվարկած *θ0,θ1, … θn* արժեքները, ընդհամենը պետք է հաշվի *hθ(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 միջին տարբերություն:

Ահա բանաձևը՝

Ավելի պարզ այն կարող ենք գրել հետևյալ կերպ՝ , որտեղ *-ը* *-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, քանզի դրա միջոցով *-ու* ածանցումից ստացված *2* բազմապատիկը կվերանա։

Ստացվեց, որ մեր խնդիրը կայանում է *J(θ0,θ1, … θn)-ը* մինիմիզացնելու մեջ, որն ավելի ֆորմալ կարող ենք ներկայացնել հետևյալ կերպ՝

**Նվազող գրադիենտ (Gradient Descent)**

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

Դիտարկենք հիպոթեզ ֆունկցիայի պարզեցված օրինակ, որն ունի ընդհամենը 2 պարամետր՝ *θ0* և *θ1*։ 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   
*θ0-ի* և *θ1-ի* արժեքների համար ինչքանով է հիպոթեզը շեղված սպասվելիք արժեքներից։ Հասկանալի է, որ պետք է գտնել տվյալ գրաֆիկի վրայի ամենացածր կետը, որի *θ0* և *θ1* արժեքներն էլ հենց կլինեն մեր հիպոթեզի որոնելի պարամետրերի արժեքները (վերևի նկարում կարմիր սլաքներով նշված են տվյալ գրաֆիկի մինիմումները)։ Քանզի 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 արագություն (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: Քայլի ուղղությունը, որոշվում է *J(θ0, θ1)-ի* մասնակի ածանցյալով: Կախված այն բանից, թե որտեղից ենք սկսում է 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 հետևյալն է․   
կրկնել հեևյալը մինչև զուգամիտում՝ որտեղ`   
*j = 0,1, … 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 *θ*0, *θ*1*, … θn* պարամետրերը: Այսինքն նախ տվյալ իտերացիայի համար հաշվարկել բոլոր *θ-ների* արժեքները՝ *θ',* որից հետո *θ-ին* վերագրել *θ'։* Եթե կամայական *θj-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 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՝

ապա թարմացման օրենքի մեջ *J(θ)-ի* արժեքը տեղադրելուց հետո թարմացման կանոնի տեսքը կլինի՝

որտեղ՝ j≔0…n:

Այստեղ և հետագայում կընդունենք, որ , բոլոր *i-երի* համար: Սա արվում է բանաձևերը հարմար ներկայացնելու համար։ Ստացվեց, որ գծային հիպոթեզն ունի հետևյալ տեսքը՝

որտեղ, ինչպես նշվեց, ։

**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, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. 2016