

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Մաթեմատիկայի և մեխանիկայի ֆակուլտետ
Ֆինանսական մաթեմատիկայի ամբիոն

ԱՎԱՐՏԱԿԱՆ ԱՇԽԱՏԱՆՔ

Թեմա՝ Բաժնետոմսի գնի փոփոխության կանխատեսումը
առքուվաճառքի առաջարկներով
Predicting price movement from limit order book

Ուսանող՝ Գուրգեն Բլբուլյան, IV կուրս, 406 խումբ
Ղեկավար՝ ֆ.մ.գ.թ. Միքայել Պողոսյան
Գրախոս՝ ֆ.մ.գ.թ. Վահագն Սաղաթելյան

Երաշխավորվել է պաշտպանության
«26» 05.2020թ.
Ամբիոնի վարիչ՝ ֆ.մ.գ.թ. Միքայել Պողոսյան

ԵՐԵՎԱՆ 2020

Նախաբան

Աշխատանքի ընթացքում կառուցվել են ներդրոնային ցանցերի CNN, LSTM, CNN-LSTM մոդելներ, որոնց միջոցով կանխատեսվել է բաժնետոմսի գնի փոփոխության ուղղությունը հաջորդ ժամանակաշրջաններում՝ օգտագործելով բաժնետոմսերի առքուվաճառքի առաջարկների տվյալները (Limit Order Book): Կանխատեսման ճշգրտությունը հասել է միջև 54%-ի, որը համեմատական է միջև այժմ եղած նմանատիպ մոդելներին:

Բովանդակություն

Նախաբան	2
Բովանդակություն	3
Ներածություն	4
1. Խնդրի դրվածքը	6
2. Տվյալների նկարագրություն	6
3. Խորացված ուսուցում	8
3.1 Gradient decent	9
4. Multilayer perceptron	13
4.1 Activation functions	14
5. Նեյրոնային ցանցերի նպատակը	15
6. Vanishing and exploding gradient	17
7. Model preformance	18
7.1 Confusion matrix	20
8. Convolutional Neural Networks (CNN)	21
8.1 Edge detection	22
8.2 Convolution operation modifications	23
8.5 Multilayer CNN	25
8.6 Pooling layer	27
9. Ռեկուրենտ նեյրոնային ցանցեր	29
9.1 Long Short Term Memory networks	32
10. Տվյալների նախապատրաստում	35
10.1 Տվյալների պիտակավորում	36
11. CNN մոդելի արդյունքը	39
12. LSTM մոդելի արդյունքը	41
13. CNN-LSTM մոդել արդյունքը	43
14. Համեմատություն այլ մոդելների հետ	45
Եզրակացություն	46
Գրականության ցանկ	47
Հավելված	50

Ներածություն

Գրեթե բոլոր ժամանակներում և ոլորտներում արդիական է այն խնդիրը, թե ինչ է սպասվում ապագայում և թե ինչպես այդ տեղեկությունը օգտագործելով նվազեցնել ռիսկը կամ ավելացնել եկամուտները:

Արդի տեխնոլոգիաները հնարավորություն են տալիս հավաքագրել մեծածավալ և բարձր հաճախությամբ տվյալներ և այդ տվյալները վարկյանների ընթացքում փոխանցել մարդկանց, որոնք օգտագործելով այդ կատեգորիաները որոշակի արժեք ապագայում:

Ժամանակակից աշխարհում ֆինանսական շուկաները հանդիսանում են միլիոնավոր մարդկանց եկամտի աղբյուր: Ապագայի մասին բոլորից առավել ինֆորմացիա ունենալու շնորհիվ հնարավոր է վաստակել մեծ գումարներ վարկյանների ընթացքում: Այս պատճառով ժամանակի ընթացքում ֆինանսական տվյալների վերլուծության բազմապիսի մոդելներ են ստեղծվել:

Ներկայումս լայնորեն կիրառվում են մեքենայական ուսուցման ալգորիթմները, որոնք հնարավորություն են տալիս բացահայտելու այնպիսի թաքնված երևույթներ, որոնք միջև այժմ եղած ալգորիթմների միջոցով հնարավոր չէր հայտնաբերել: Հետազոտության ընթացքում օգտագործվել է բաժնետոմսի առքուվաճառքի առաջարկների տվյալներ (Limit Order Book), որոնք այժմ լայնորեն կիրառվում են բաժնետոմսի առքուվաճառքով զբաղվող անձանց կողմից:

Հետազոտության ընթացքում օգտագործվել են 3 մոդելներ՝ CNN, LSTM և CNN-LSTM, որոնցից վերջինը ժամանակային շարքերի համար հիմնականում օգտագործվում է մարդկային շարժումները դասակարգելու համար: Համացանցում կան մի քանի հոդվածներ, որոնք կիրառում են այս մոդելները LOB տվյալների վրա (1, 2): Այս աշխատանքի մեջ կառուցվել են ավելի պարզ մոդելներ, որոնցով ստացված արդյունքները կարող են մրցակցել վերոնշյալ մոդելների արդյունքների հետ: Աշխատանքի ընթացքում կիրառվել են տվյալների պիտակավորման մի քանի մեթոդներ, որոնք ևս տարբերվում են վերոնշյալ մոդելներից: Հիմնականում օգտագործվել է մարդկային շարժման

դասակարգման մասին գրականությունը՝ հասկանալու համար մոդելների
յուրահաստկությունները:

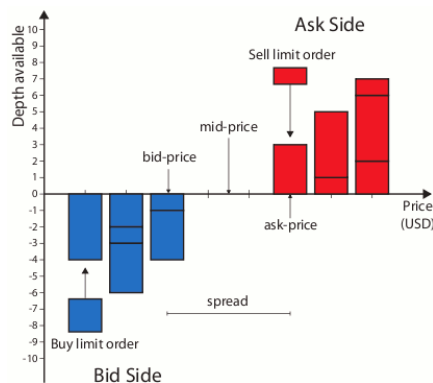
Աշխատանքի ընթացքում փորձ է արվել մոդելները ներկայացնել այնպես,
որ ռեգրեսիայի մասին նախնական գիտելիքներ ունեցող անձը հասկանա
հետազոտությունը ամբողջությամբ, այդ իսկ պատճառով օրինակները բերված
են պարզ խնդիրների վրա:

1. Խնդրի դրվածքը

Աշխատանքի հիմնական նպատակն է կառուցել այնպիսի մոդելներ, որոնց միջոցով հնարավոր կլինի կանխատեսել բաժնետոմսի գնի ապագա տեղաշարժի ուղղությունը՝ օգտագործելով Limit Order Book տվյալներ: Ապագա տեղաշարժի ուղղությունն ասելով հասկանում ենք, թե արդյոք մոտ ապագայում բաժնետոմսի գինը կաճի, կնվազի թե կմնա հաստատուն:

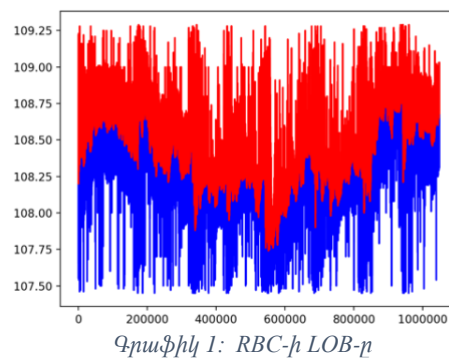
2. Տվյալների նկարագրություն

Նախ հասկանանք ինչ է իրենից ներկայացնում Limit Order Book-ը (LOB): Դիցուք ունեն բարձր իրացվելի բաժնետոմս, և յուրաքանչյուր վայրկյան որոշակի մարդիկ պատրաստ են վաճառել իրենց բաժնետոմսերը և որոշներն էլ պատրաստ են գնել դրանք: LOB-ը իր մեջ ներառում է բոլոր գնելու և վաճառելու առաջարկները (orders): Այն կազմված է երկու մասից Ask և Bid: Ask մասում գտնվում են բոլոր այն order-ները որոնք կատարվել են վաճառողների կողմից, այսինքն թե ինչքան գնով և քանի հատ բաժնետոմս են պատրաստ վաճառել այդ պահին: Bid մասում գտնվում են բոլոր այն order-ները որոնք կատարվել են գնորդների կողմից, այսինքն թե ինչքան գնով և քանի հատ բաժնետոմս են պատրաստ գնել դրանք: Ամենամեծ գնով bid order-ը և ամենափոքր գնով ask order-ը կոչվում են level 1 LOB: Եթե մեր տվյալները պարունակում են օրինակ 3 ամենամեծ bid order-ները և 3 ամենափոքր ask order-ները կասենք որ ունենք level 3 LOB: Յուրաքանչյուր պահին հնարավոր է ունենալ բազմաթիվ order-ներ, և նույն տրամաբանությամբ LOB տվյալների level-ը կախված է թե քանի տվյալ է պարունակում այն իր մեջ:



Հետազոտության ընթացքում օգտագործվել է Royal Bank of Canada-ի բաժնեթղթերի LOB-ը, որտեղ ներառված էին բոլոր այն order-ները որոնք արվել էին 2019թ-ի նոյեմբերի 7-ից 18-ը ընկած ժամանակահատվածում, և վերցվել են միայն այն order-ները որոնք արվել են այն ժամանակ, երբ շուկան բաց էր, որպեսզի միայն նորմալ առևտրային գործարքները ներառվեն:

Տվյալները բազմությունը ներառում է 1048575 տվյալ, որոնցից յուրաքանչյուրը իրենից ներկայացնում է մեկ order և դասավորված են ժամանակային հաջորդականությամբ: Տվյալները ունեն միլիվայրկյանական հաճախականություն: Գրաֆիկ 1-ը ցույց է տալիս ամեն պահին եղած bid և ask order-ների գները:



Ինչու՞ LOB

Ինչպես գիտենք բաժնետոմսի գնի վրա ազդող հիմնական գործոններից են առաջարկը և պահանջարկը: LOB-ի միջոցով հնարավոր է ամեն պահին գնահատել առաջարկը և պահանջարկը և դրա միջոցով որոշում կայացնել գնել թե վաճառել բաժնետոմսը:

3. Խորացված ուսուցում

Խորացված ուսուցումը ավելի լավ հասկանալու համար, հարմար է սկսել պարզ լոգիստիկ ռեգրեսիայից: Դիցուք ունենք $(x^{(i)}, y^{(i)})$ զույգեր,

$$\text{որտեղ } x^{(i)} \in R^{n_x} \text{ և } y^{(i)} \in \{0,1\}, \forall i \in [1, \dots, m]:$$

$x^{(i)}$ -երը ամեն մի տվյալի բնութագրիչների վեկտորներն են, $y^{(i)}$ -երը ցույց են տալիս թե ամեն մի տվյալ որ դասին է պատկանում: Ունենալով այս տվյալները ցանկանում ենք կառուցել մոդել, որը մեզ կվերադարձնի $\hat{y} = P(y = 1|x)$, ցանկացած նոր տվյալի համար: Սովորական ռեգրեսիայի պարագայում չենք կարող \hat{y} -ը դիտարկել որպես հավանականություն, որովհետև \hat{y} -ը կարող է մեծ լինել 1-ից կամ փոքր 0-ից:

$$\hat{y} = w^T x + b \quad w \in R^{n_x}, b \in R$$

Որպեսզի այս խնդիրը լուծվի, լոգիստիկ ռեգրեսիայի պարագայում

$$\hat{y} = \sigma(w^T x + b), \quad \text{որտեղ } \sigma(z) = \frac{1}{1 + e^{-z}}$$

Ունենալով m հատ տվյալ, մենք ցանկանում ենք գտնել այնպիսի w վեկտոր, որի պարագայում

$$\hat{y}^{(i)} \approx y^{(i)} \quad \forall i \in [1, m]$$

Այս խնդիրը լուծելու համար մեզ անհրաժեշտ է ֆունկցիա, որը կչափի թե ինչքան լավն է \hat{y} -ը երբ իրական պիտակը y -ն է: Այդ ֆունկցիան անվանենք

$$\text{Loss function} = L(\hat{y}, y)$$

Լոգիստիկ ռեգրեսիայի պարագայում հիմնականում վերցնում են հետևյալ Loss function-ը:

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

Մենք կարող ենք պայմանական հավանականությունը ներկայացնել այս կերպ, եթե $y=1$ մենք ստանում ենք \hat{y} , իսկ եթե $y=0$ ՝ $1-\hat{y}$

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

Մենք ցանկանում ենք որպեսզի հնարավորինս մեծ լինի այս հավանականությունը:

$$\text{Գիտենք որ } \operatorname{argmax} p(y|x) = \operatorname{argmax} \log(p(y|x))$$

ԵՎ եթե վերցնենք բացասական նշանով կստանանք loss function-ը:

$$\log p(y|x) = \log [\hat{y}^y (1 - \hat{y})^{1-y}]$$

$$-\log p(y|x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Ինչպես երևում է ներքևում երբ $y=1$ մենք ցանկանում ենք որ \hat{y} -ը լինի մեծ, և երբ $y=0$ \hat{y} -լինի փոքր՝

$$L(\hat{y}, y) = \begin{cases} -\log(\hat{y}), & \text{if } y = 1 \\ -\log(1 - \hat{y}), & \text{if } y = 0 \end{cases}$$

Բազմադաս դասակարգման խնդրում loss function-ը ունի հետևյալ տեսքը: Հաճախ այն անվանում են նաև cross-entropy loss:

$$L(\hat{y}, y) = - \sum_{j=1}^k y_j \log(\hat{y}_j)$$

Մենք հաշվելով loss function-ը յուրաքանչյուր օրինակի համար և այդ արդյունքները միջինացնելով կստանանք cost function:

Ինչպես տեսնում ենք cost function -ը կախված է պարամետրերից, և մենք

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$$

պետք է գտնենք այն պարամետրերը, որի արդյունքում cost functiony կլինի հնարավորինս փոքր:

3.1 Gradient decent

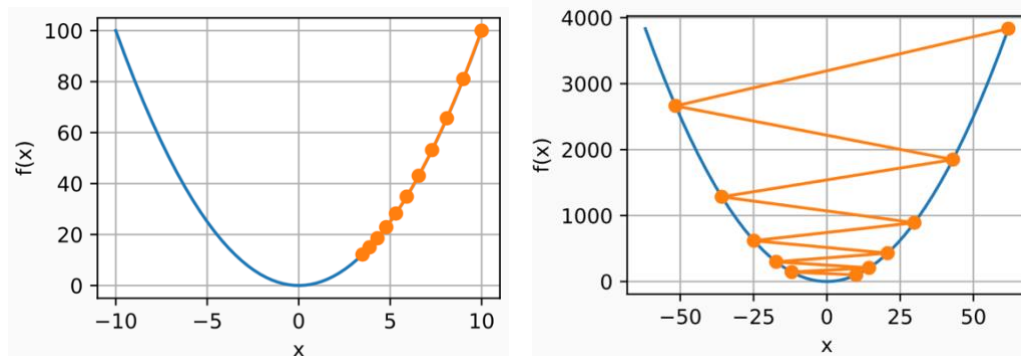
Լոջիստիկ ռեգրեսիայի պարագայում Cost functiony մինիմալացնելու համար, կարող ենք իհարկե ածանցել այն և գտնել մինիմում արժեքը հեշտորեն, քանի որ այն ուռուցիկ ֆունկցիա է և ունի ընդամենը մեկ մինիմումի կետ: Բայց հետագայում կտեսնենք որ խորացված ուսուցման ժամանակ հաճախ հանդիպում են այնպիսի cost function-ներ որոնք ունեն հարյուրավոր լոկալ և գլոբալ մինիմումի կետեր, և հաճախ անհնար է տալ դրանց մաթեմատիկական արտապատկերումը ֆունկցիայի տեսքով: Այդ պատճառով անհրաժեշտ է մեկ այլ բազմաֆունկցիոնալ գործիք: Այն ինչ անում է gradient decent-ը հետևյալն է: Վերցնելով սկզբնական արժեքներ պարամետրերի համար հաշվում է cost function-ի արժեքը: Այնուհետև հաշվելով cost function-ի

ածանցյալը ըստ պարամետրի, թարմացնում է պարամետրի արժեքը հետևյալ բանաձևով:

$$\begin{aligned} w_j &= w_j - \alpha \frac{\partial}{\partial w_j} J(w, b) \\ b &= b - \alpha \frac{\partial}{\partial b} J(w, b) \end{aligned} \quad (\text{for } j = \overline{1, m})$$

Այստեղ α -ն անվանում ենք learning rate: α -ն ցույց է տալիս թե ինչքան արագ է ալգորիթմը թարմացնում պարամետրերը:

Գրաֆիկորեն պատկերը հետևյալն է:

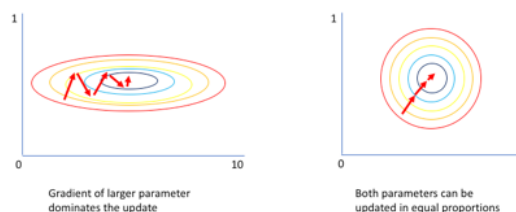


Գրաֆիկում երևում է, որ ամեն մի քայլում ձգտում ենք դեպի օպտիմումը և եթե մեծ վերցնենք α -ն հնարավոր է որ օպտիմումին ձգտենք ավելի քառասային շարժումներով:

Այսինքն α -ն շատ կարևոր հիպերպարամետր է gradient decent-ի համար: Հաճախ կիրառում են learning rate decay տեխնիկան, որի միջոցով ամեն քայլում ինչ-որ չափով α -ն փոքրացնում են:

Հաճախ տվյալներ լինում են տարբեր չափողականության, դրա համար անհրաժեշտ է տվյալները բերել նույն չափողականության, որպեսզի ավելի արագ լինի սովորելու գործընթացը:

Why normalize?



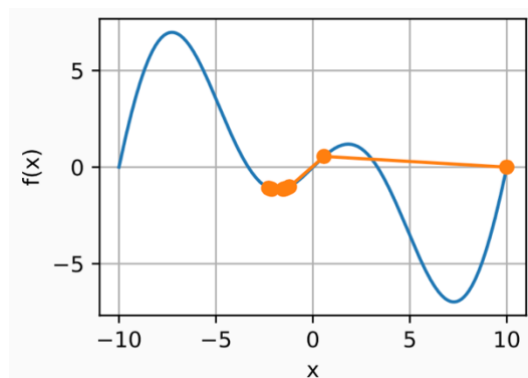
Տարբերում են մի քանի ալգորիթմներ:

Batch gradient decent

Եթե ամեն քայլում cost function-ը հաշվենք բոլոր տվյալները հաշվի առնելով և պարամետրերը փոխենք դրա հիման վրա, ապա մենք սահուն կերպով կմոտենանք լոկալ մինիմումին: Բայց կան մի քանի այլ ալգորիթմներ, որոնց շնորհիվ հնարավորություն է ստեղծվում նույնիսկ ձգտել գլոբալ մինիմումին:

Mini batch gradient decent

Այս ալգորիթմի ժամանակ ամեն քայլը կատարվում է տվյալներ որոշակի խմբի հիման վրա: Օրինակ եթե 200 տվյալ ունենք, կարող ենք այն պատահականորեն բաժանել 10 մասի և ամեն քայլում 20 տվյալի համար հաշվել cost function-ը և ըստ դրա փոխել պարամետրերը: Այս ալգորիթմի միջոցով ավելի քառսային ենք մոտենում օպտիմումին և հնարավորություն է ընձեռնվում լոկալ մինիմումից տեղափոխվել դեպի գլոբալ մինիմում: Այս ալգորիթմը ավելի արագ է բայց նաև անկանխատեսելի որոշ չափով:

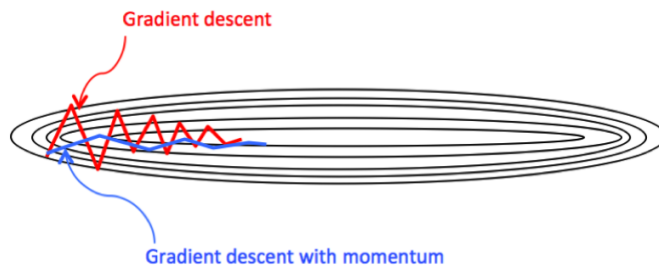


Stochastic gradient decent

Այս ալգորիթմի ժամանակ փոփոխությունը կատարվում է միայն մեկ տվյալ հաշվի առնելով: Այս ալգորիթմը ավելի քառսային է ձգտում օպտիմումին, ավելի արագ է և հնարավորություն է տալիս լոկալ մինիմումից տեղափոխվել գլոբալ մինիմումի:

Gradient decent with momentum

Այս ալգորիթմի ժամանակ որոշակիորեն հարթեցվում է: Պարամետրերը փոխվում են հաշվի առնելով նախորդ փոփոխությունները միջինացնելով: Սրա շնորհիվ ավելի քիչ քառասային շարժումով ենք ձգտում օպտիմումին:



RMSprop

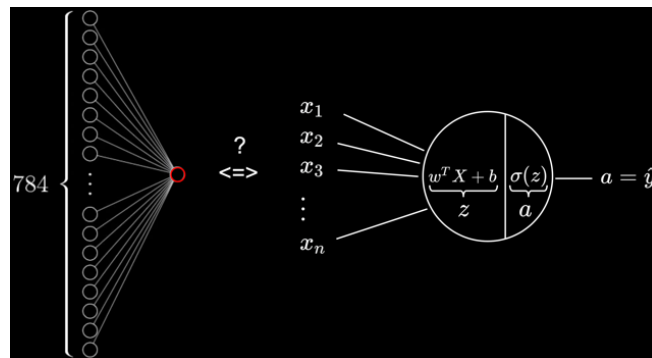
Այս ալգորիթմի ժամանակ հաշվի են առնվում cost function-ի փոփոխությունը բոլոր առանցքներով, և եթե շատ է տատանվում ըստ x առանցքի օրինակ, ապա ալգորիթմը փորձում է փոքրացնել այդ տատանումը:

Adam

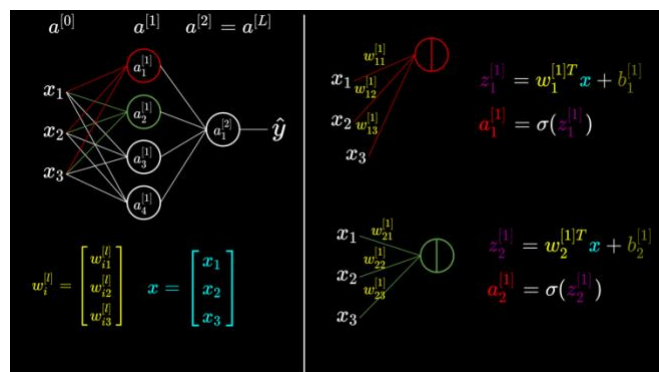
Այս ալգորիթմը RMSprop-ի և gradient decent with momentum-ի զուգակցությունն է:

4. Multilayer perceptron

Լոգիստիկ ռեգրեսիան կարելի է ներկայացնել որպես մեկ նեյրոն ունեցող նեյրոնային ցանց: Օրինակ եթե ունենք նկարների որոնք 784 պիքսել ունեն, և դրանք եթե համարենք որպես ամեն նկարի բնութագրիչներ, ապա լոգիստիկ ռեգրեսիա միջոցով կարող ենք կանխատեսել արդյոք նոր նկարը կատու է թե ոչ:



MLP նեյրոնային ցանցերը ունեն հետևյալ տեսքը:



Ինչպես տեսնում ենք նեյրոնային ցանցը իրենից ներկայացնում է մի քանի լոգիստիկ ռեգրեսիաների համադրություն: Երկրորդ շերտում, որը նաև անվանում են գաղտնի շերտ, մենք տարբեր պարամետրերով կիրառում ենք 4 լոգիստիկ ռեգրեսիա, և այդ 4 նոր արդյունքները համարելով որպես նոր բնութագրիչներ ամեն տվյալի համար, կատարում ենք ևս մեկ լոգիստիկ ռեգրեսիա, որի արդյունքում ստանում ենք վերջնական արդյունք: Ցանցերը կարող են ունենալ բազմաթիվ շերտեր և նեյրոններ:

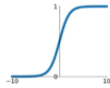
4.1 Activation functions

Նեյրոնային ցանցերի մեջ sigmoid ֆունկցիայի փոխարեն հաճախ կիրառում են այլ ֆունկցիաներ, որոնք կոչվում են activation ֆունկցիաներ: Ընդվորում ամեն շերտում կարող են տարբերվել ֆունկցիաները:

Activation Functions

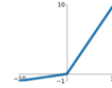
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



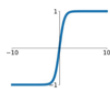
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

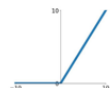


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

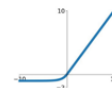
ReLU

$$\max(0, x)$$

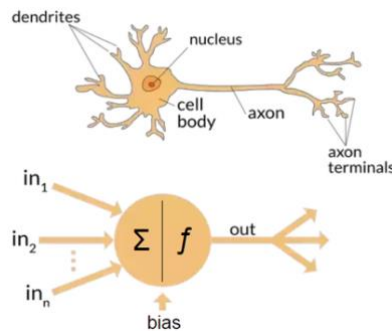


ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation ֆունկցիաների միջոցով որոշվում է թե որ նեյրոններն են սկսում գործել: Մարդկային ուղեղի նեյրոնային նման, արհեստական նեյրոնները ստանալով որոշակի ազդակներ և անցնելով որոշակի փուլով կատարում են որոշակի գործողություն: Activation ֆունկցիաների միջոցով է որոշվում, թե հաջորդ շերտում որ նեյրոնները պետք է ակտիվանան:

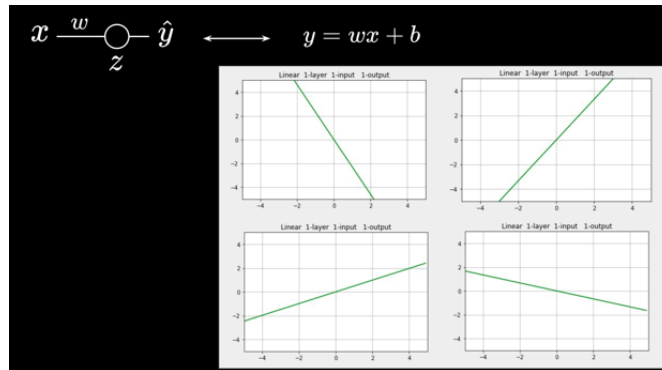


Եթե ցանկանում ենք օրինակ դասակարգել 3 դասի նկարները, ապա վերջին շերտում կարող է լինել 3 նեյրոն, որը ցույց կտա ամեն դասին պատկանելու հավանականությունը: Դա հաշվարկվում է softmax ֆունկցիայի միջոցով և վերջին շերտը կոչում ենք softmax layer:

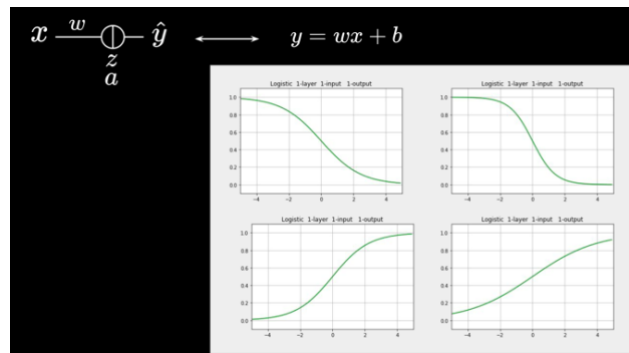
$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

5. Նեյրոնային ցանցերի նպատակը

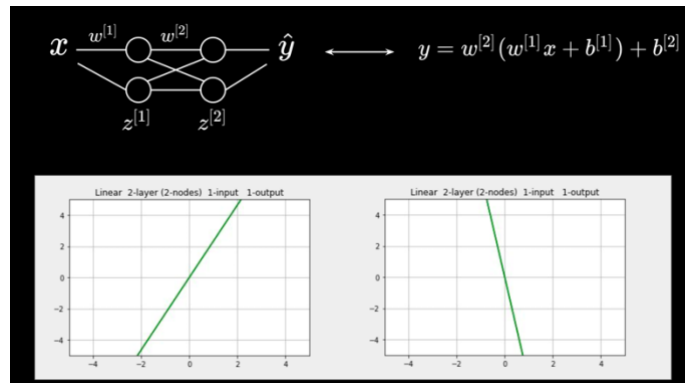
Հարց է առաջանում թե ինչի համար են պետք activation ֆունկցիաները, և բազմակի շերտերը: Պարզ գծաին ռեգրեսիայի պարագայում մենք մեր տվյալների բազմությունը փորձում ենք մոտարկել ուղիղ գծով, որը կարող ենք ներկայացնել որպես նեյրոնային ցանց առանց activation ֆունկցիայի:



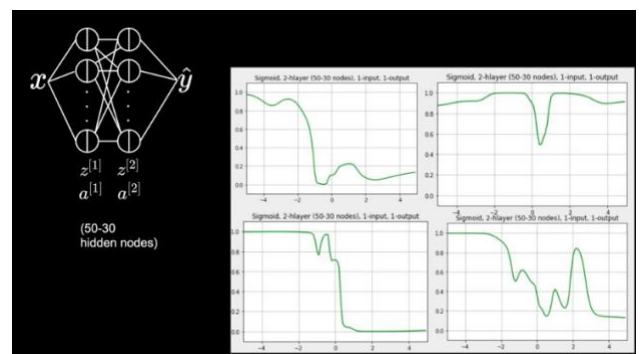
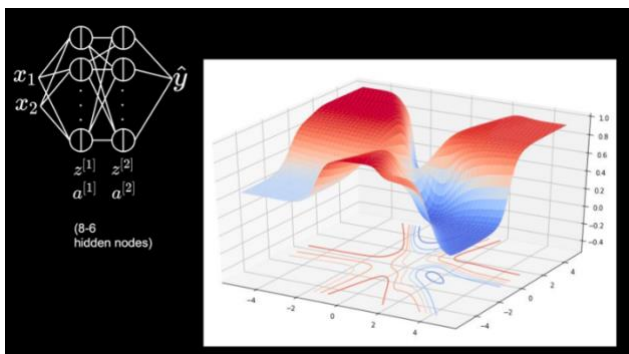
Եթե կիրառում ենք լոգիստիկ ռեգրեսիա, այսինքն activation ֆունկցիան վերցնում ենք որպես սիգմոիդ ֆունկցիա, այդ պարագայում տվյալները մոտարկում ենք սիգմոիդ ֆունկցիայով:



Եթե կառուցենք երկշերտ նեյրոնային ցանց առանց ակտիվացնող ֆունկցիայով այդ պարագայում գրեթե ոչինչ չի փոխվի, մենք տվյալները կմոտարկենք ուղիղ գծով:



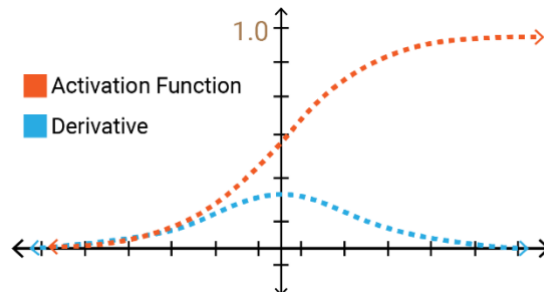
Բայց բազմաշերտ նեյրոնային ցանցում, որտեղ օգտագործում ենք activation ֆունկցիաներ, տեսնում ենք հնարավոր է ստանալ բավականին բարդ ֆունկցիաներ, որոնք ավելի մոտ են իրականությանը և ցանկացած տվյալների բազմության համար կարելի է կառուցել այնպիսի նեյրոնային ցանց, որի շնորհիվ մեր բոլոր տվյալները կանցնեն այդ ֆունկցիայով, բայց մեծ քանակի տվյալների դեպքում այդ ցանցը բավականին մեծ կլինի:



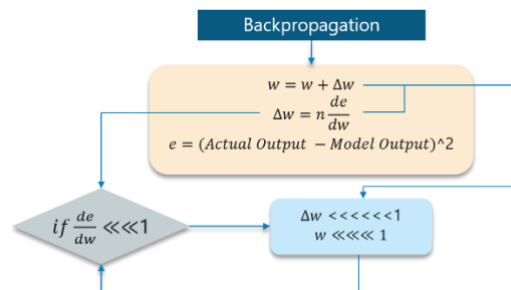
Այսպիսով activation ֆունկցիաները գծայնությունը կոտրելու համար են, և նեյրոնները քանակը և շերտերը ավելի բարդ ֆունկցիաներ ստանալու համար են:

6. Vanishing and exploding gradient

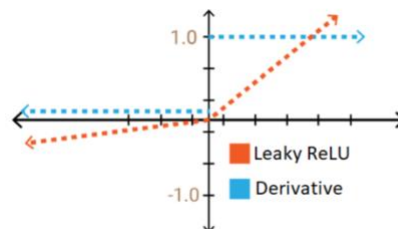
Gradient decent ալգորիթմների ժամանակ առաջանում են vanishing and exploding gradient-ի խնդիրը:



Միգմոիդ ֆունկցիայի ածանցիալը մեծ արժեքների համար ընդունում է շատ փոքր արժեք: Եթե մենք ունենանք շատ երկար նեյրոնային ցանց, ապա ժամանակի ընթացքում կստացվի, որ ալգորիթմը սկսում է պարամետրերը ավելի դանդաղ փոխել և սովորելու գործընթացը դանդաղում է:



Այս երևույթը կոչվում է vanishing gradient: Exploding gradient-ի դեպքում պարամետրերը շատ արագ են փոփոխվում, որը ևս կարող է խնդիրներ առաջացնել: Այս խնդիրը կարելի է լուծել ուրիշ activation ֆունկցիաների միջոցով: Օրինակ leaky ReLU ֆունկցիան օգտագործելով:

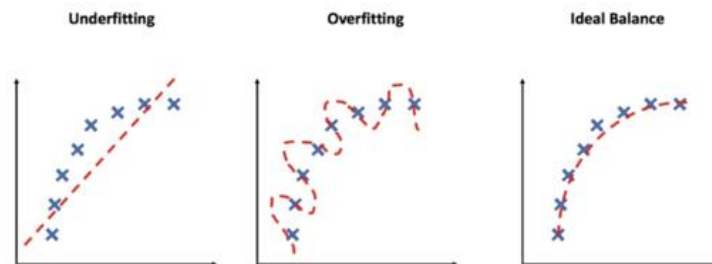


7. Model preformance



Մոդելները կառուցելիս, հաճախ ունեցած տվյալները բաժանում են 3 մասի՝ train set, validation set, test set-ի: Train set-ի վրա կառուցում ենք մոդելը, validation set-ի վրա փորձարկում ենք մոդելը ամեն քայլում իսկ test set-ի վրա փորձարկելով վերջնական մոդելը կարծիք ենք ձևավորում մոդելի որակի վրա: Հաճախ test և validation set-երը նույնանում են:



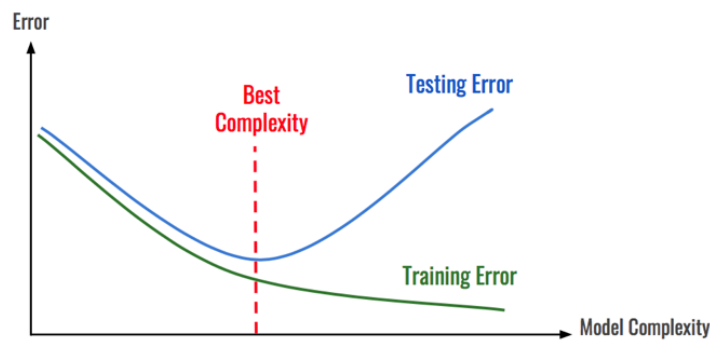
Կախված մոդելի բարդությունից՝ հնարավոր է ունենալ տարբեր արդյունքներ train և test set-երի վրա: Օրինակ եթե տվյալները դասավորված են էքսպոնենցիալ կերպով և մենք այն մոտարկում ենք ուղիղ գծով, մենք կունենանք underfitting-ի կամ high bias-ի խնդիր, այսինքն մենք տվյալների օրինաչափությունը սխալ ենք մոտարկել:



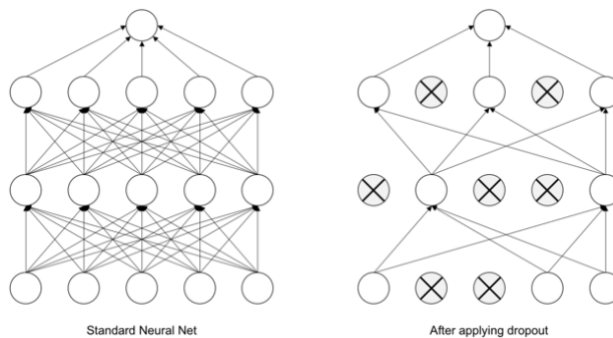
Եթե մենք այնպես կառուցենք ֆունկցիա, որը կանցնի բոլոր կետերով այս պարագայում մենք կունենանք overfitting-ի կամ high variance խնդիր: Այս պարագայում մեր ֆունկցիան կսովորի «անգիր» մեր տվյալները և նոր տվյալի համար բավականին սխալ մոտարկում կստանանք: Այսինքն մենք պետք է այնպիսի մոդելի բարդություն ընտրենք որ հնարավորինս լավագույն արդյունքները ստանանք train և test set-ի վրա:

Cat classification:		$y = 1$	$y = 0$	
				
Train set error:	1%	15%	15%	0.5%
Val set error:	11%	16%	30%	1%
	High variance	High bias	High bias & high variance	Low bias & low variance

High bias-ի խնդիրը կարելի է լուծել կառուցելով ավելի բարդ մոդելներ և ավելի երկար սովորեցնել մոդելը, այսինքն մոդելի պարամետրերը ավելի շատ անգամներ փոփոխել:



High variance-ի խնդիրը լուծելու համար օրինակ կարելի է ավելի շատ տվյալներ ավելացնել training set-ի մեջ: Կարելի է կեղծ տվյալներ ավելացնել, օրինակ նկարների պարագայում շրջել որոշակի անկյան տակ, կամ գույները խամրեցնել: Այս տեխնիկան կոչվում է data augmentation: Կարելի է նաև կատարել dropout տեխնիկան, որի ժամանակ պատահականորեն, որոշակի հավանականությամբ ջնջում ենք նեյրոններ, որի արդյունքում ունենում ենք ավելի պարզ ցանց: ԵՎ կան այլ բազմապիսի մեթոդներ:



7.1 Confusion matrix

Մոդելների արդյունքները չափելու միջանի մեթոդներ կան, որոնցից մեկը confusion matrix-ն է: Այն իրենից ներկայացնում է աղյուսակ, որը ցույց է տալիս ճշգրիտ և ոչ ճշգրիտ կանխատեսված տվյալների քանակը: Դիցուք ունենք կենդանիների նկարներ և ցանկանում ենք կառուցել մոդել, որի միջոցով կկարողանանք ճիշտ կանխատեսել կենդանիների տիպը: Մոդելը test set-ի վրա կիրառելուց հետո, կառուցենք confusion matrix-ը:

		True/Actual		
		Cat (🐱)	Fish (🐟)	Hen (🐔)
Predicted	Cat (🐱)	4	6	3
	Fish (🐟)	1	2	0
	Hen (🐔)	1	2	6

Ինչպես տեսնում ենք, անկյունագծի թվերը ցույց են տալիս այն տվյալների քանակը, որոնք կանխատեսվել է ճիշտ: Եթե վերցնենք առաջին սյունը կտեսնենք որ 4 կատվի նկար ճիշտ է դասակարգել մոդել և մեկական նկարներ այն սխալմամբ դասակարգել է որպես ձուկ և հավ:

Confusion matrix-ի հիման վրա կարելի է հաշվել միջանի ցուցանիշներ, որոնք պատկերացում կտան մոդելի որակի մասին: Դրանցից են precision-ը և recall-ը: Այն հաշվում են ամեն մի դասի համար առանձին: Օրինակ կատուների դասի համար precision-ը հավասար է $\frac{4}{4+6+3}$, որը ցույց է տալիս կատուների ճիշտ կանխատեսված տվյալների և կատու կանխատեսված տվյալների քանակների հարաբերությունն է: Կատուների դասի համար recall-ը հավասար է $\frac{4}{4+1+1}$, որը կատուների ճիշտ կանխատեսված տվյալների և իրական կատուների տվյալների քանակների հարաբերությունն է:

F1-score-ը precision-ի և recall-ի հարմոնիկ միջինն է, որը հաշվում են հետևյալ կերպ՝

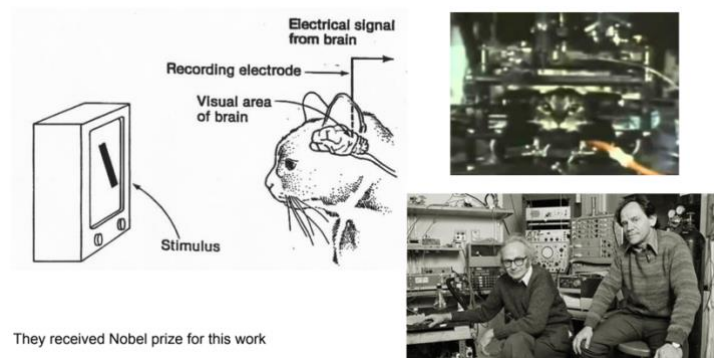
$$F1\ score = 2 \times \frac{precision \times recall}{precision + recall}$$

8. Convolutional Neural Networks (CNN)

CNN-ը կիրառվում է այնպիսի ոլորտներում, ինչպիսիք են ինքնակառավարվող մեքենաները, անվտանգության, ֆիլմարտադրություն, առողջապահություն, ֆինանսներ և բազմապիսի այլ ոլորտներում:

CNN-ի գախափարը սկսվել է 1960-ականների մի դաժան փորձից ([experiment video](#)), որտեղ Հուբելը և Վիեսելը կատարում էին փորձ կատվի վրա, որի համար հետագայում ստացան նոբելյան մրցանակ:

Hubel and Wiesel (1959 - 1968) - experiment with cat

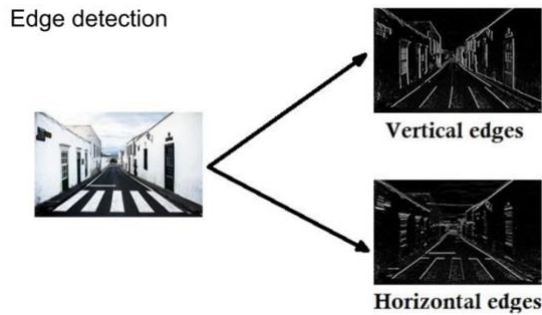


They received Nobel prize for this work

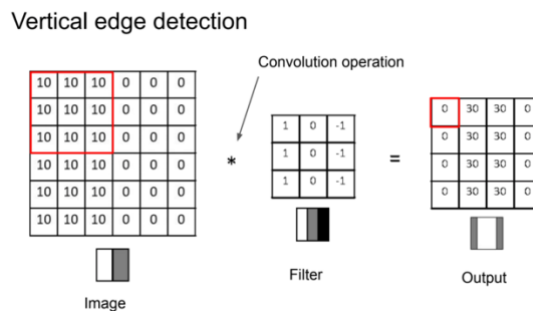
Նրանք, տեղադրելով էլեկտրոդներ կատվի ուղեղի նեյրոնների վրա, կատվին ցույց էին տալիս պատկերներ ստվերներ, փորձում էին հասկանալ, թե ինչպես է արձագանքում ուղեղը: Փորձելով տարբեր պատկերներ, կատվի ուղեղի նեյրոնը ոչ-մի իմպուլս չէր հաղորդում: Որոշ ժամանակ անց նրանք նկատեցին, որ նեյրոնը արձագանքում է այն ժամանակ, երբ նրանք թիթեղը հանում են լուսարձակիչի միջից: Մի շարք փորձերից հետո նրանք եկան այն եզրահանգման, որ նրանք դիպել էին այն նեյրոնին, որը պատասխանատու էր գծերը ընկալելու համար: Երբ նրանք հանում էին թիթեղը, թիթեղի եզրը ուղիղ գիծ էր ստվերում կատվի աչքին, և այդ պատճառով կատվի այդ նեյրոնը արձագանքում էր պատկերին: Հետագայում պարզ դարձավ, որ ամեն մի նեյրոն պատասխանատու է ավելի պարզ դետալներ ընկալելու համար: Եվ այդ պարզ դետալները համախմբելով՝ մենք ընկալում ենք որևէ առարկա ամբողջությամբ:

8.1 Edge detection

Մարդիկ փորձեցին այս գաղափարի շուրջ մաթեմատիկական մոդելներ կառուցել, որը որոշակի հաջողության հասավ նկարների մեջ եզրերը գտնելու հարցում: Ցանկացած եզր կարելի է դիտարկել, որպես մուգ գույնից դեպի բաց գույնի անցում կամ հակառակը:



Տեսնենք թե ինչպես կարելի է դա անել պարզ մաթեմատիկական գործողությունների արդյունքում:



Դիցուք ունենք քառակուսի, որի աջ կողմը սպիտակ է և ձախ կողմը մոխրագույն: ԵՎ քանի որ ունենք բաց գույնից դեպի մուգ գույն անցում, ուրեմն մենք ունենք եզր այդ պատկերում, որը գտնվում է անմիջապես մեջտեղում: Գիտենք որ ցանկացած գույն կարելի է ստանալ կարմիր, կանաչ և կապույտ գույների համադրությամբ, և այդ գույները կարելի է ներկայացնել երեք թվի համադրությամբ: Այս պարագայում ենթադրենք որ ունենք մեկ գույն և ինչքան փոքր է թիվը, այնքան այն ավելի մոտ է սևին, և հակառակը ինչքան մեծ է այնքան մոտ է սպիտակին: Այնուհետև սահմանում ենք որոշակի ֆիլտեր: Ֆիլտերը տեղադրելով ձախ անկյունում, համապատասխան ֆիլտրի արժեքները բազմապատկում ենք նկարի տվյալ վայրում գտնվող արժեքով, և այդ արժեքները գումարում ենք իրար (այսուհետև convolution operation): Այնուհետև սահեցնելով ֆիլտերը նկարի վրայով կատարում ենք նույն

գործողությունները: Այդ արդյունքները հերթականությամբ դասավորելով նոր մատրիցի մեջ, ստանում ենք նոր պատկեր, որի միջոցով հասկանում ենք թե արդյոք եզր կա նկարում, թե ոչ: Վերևում բերված օրինակում տեսնում ենք, որ այս ֆիլտրերը միայն գտնում է ուղահայաց գծերը, բայց կան նաև հորիզոնական, անկյունային և այլ տիպի գծեր: Ժամանակի ընթացքում իհայտ էին գալիս բազմապիսի այլ ֆիլտրեր որոնք ավելի բարդ էլեմենտներ էին բացահայտում նկարի մեջ:

Prewitt masks

<table><tr><td>1</td><td>0</td><td>-1</td></tr><tr><td>1</td><td>0</td><td>-1</td></tr><tr><td>1</td><td>0</td><td>-1</td></tr></table>	1	0	-1	1	0	-1	1	0	-1	Vertical	<table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>-1</td><td>-1</td><td>-1</td></tr></table>	1	1	1	0	0	0	-1	-1	-1	Horizontal
1	0	-1																			
1	0	-1																			
1	0	-1																			
1	1	1																			
0	0	0																			
-1	-1	-1																			
<table><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>-1</td><td>0</td><td>1</td></tr><tr><td>-1</td><td>-1</td><td>0</td></tr></table>	0	1	1	-1	0	1	-1	-1	0	Diagonal (135 degrees)	<table><tr><td>-1</td><td>-1</td><td>0</td></tr><tr><td>-1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr></table>	-1	-1	0	-1	0	1	0	1	1	Diagonal (45 degrees)
0	1	1																			
-1	0	1																			
-1	-1	0																			
-1	-1	0																			
-1	0	1																			
0	1	1																			

8.2 Convolution operation modifications

Գոյություն ունեն convolution operation-ի մի քանի ձևափոխություններ, որոնք հնարավորություն են տալիս լուծել որոշակի խնդիրներ, որոնք առաջանում են convolution operation-ի ժամանակ: Դիցուք ունենք 3x3 չափի նկար, կիրառելով 2x2 չափի ֆիլտրեր մենք կստանանք 2x2 չափի նոր մատրից:

Input		Kernel		Output																	
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	*	<table><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table><tr><td>19</td><td>25</td></tr><tr><td>37</td><td>43</td></tr></table>	19	25	37	43
0	1	2																			
3	4	5																			
6	7	8																			
0	1																				
2	3																				
19	25																				
37	43																				

$$\begin{aligned} 0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19, \\ 1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25, \\ 3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37, \\ 4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43. \end{aligned}$$

Ընդհանուր դեպքում, եթե ունենք $n_h \times n_w$ չափի նկար, և կիրառում ենք $k_h \times k_w$ չափի ֆիլտրեր, ստացվող մատրիցը կլինի հետևյալ չափի՝

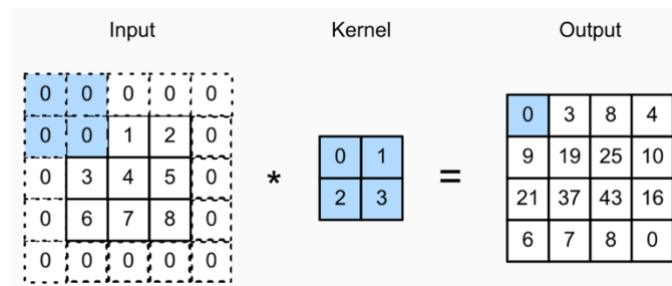
$$(n_h - k_h + 1) \times (n_w - k_w + 1)$$

Ինչպես տեսնում ենք convolution operation կիրառելով ստացվող մատրիցները սկսում են փոքրանալ: Հաջորդ խնդիր այն է երբ եզրային պիքսելները ավելի քիչ են ներառվում հաշվարկների մեջ քան կենտրոնական

պիքսելներ, մյուսը որ այս գործողությունները կարող են շատ երկար ժամանակ պահանջել հաշվելու համար:

Padding

Առաջին երկու խնդիրները մասնակիորեն լուծվում են padding-ի միջոցով, նկարի բոլոր կողմերից ավելացնելով զրոներ, և կիրառելով convolution operation-ը կունենաք, որ եզրային գծերը ավելի շատ են հաշվարկի մեջ ներառվում և նկարը համեմատաբար ավելի քիչ չափով է փոքրանում:

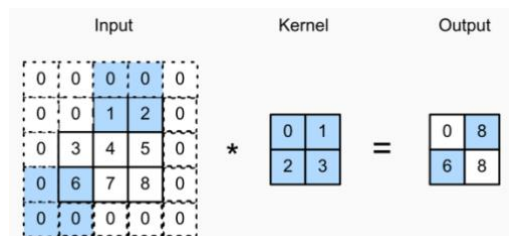


Եթե մենք ավելացնենք p_h - ական տող վերևում և ներքևում և p_w -ական սյուն աջ մասում և ձախ մասում, ապա ընդհանուր դեպքում մեր ստացվող մատրիցը կլինի հետևյալ չափի՝

$$(n_h + 2p_h - k_h + 1) \times (n_w + 2p_w - k_w + 1)$$

Stride

Երրորդ խնդիրը մասնակիորեն լուծվում է stride-ի միջոցով: Stride-ի ժամանակ ֆիլտրերը տեղաշարժում ենք ոչ թե մեկ քայլով դեպի աջ կամ ներքև, այլ մեկից ավելի քայլով: Սա հեշտացնում է հաշվարկները և նաև լուծում է այն խնդիրը որ մեկ պիքսելը կարող է շատ անգամներ հաշվարկվի, որը հավելյալ ժամանակի կորուստ է:

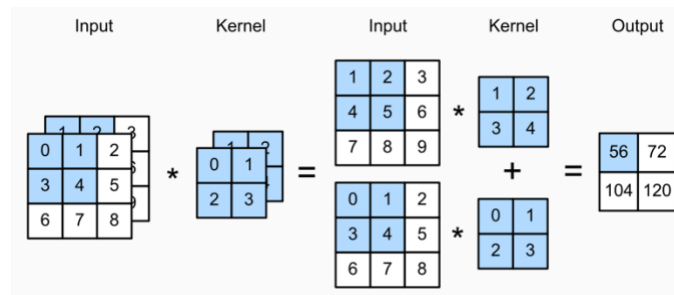


Այս պարագայում եթե s_h քայլով դեպի ներքև տեղաշարժենք ֆիլտրերը և s_w քայլով դեպի աջ, ապա ստացվող մատրիցը կլինի հետևյալ չափի՝

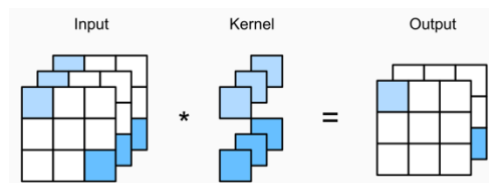
$$\frac{(n_h + 2p_h - k_h + 1)}{s_h} \times \frac{(n_w + 2p_w - k_w + 1)}{s_w}$$

8.5 Multilayer CNN

Այժմ ենթադրենք ունենք նկար, որը կազմված է երկու գույնից, այսինքն մեր նկարը օրինակ կապույտի և կարմիրի համադրությամբ է։ Այս պարագայում նկարի համար ավելանում է ևս մեկ չափողականություն, որը ցույց է տալիս նկարի շերտերի քանակը։ Այս պարագայում ֆիլտրի չափողականությունը ևս ավելանում է մեկով, որը կանվանենք՝ ֆիլտրի խորություն։ Ֆիլտրի խորությունը միշտ պետք է հավասար լինի նկարի շերտերի քանակին։ Օրինակ ունենք $3 \times 3 \times 2$ չափի նկար, ֆիլտրը $2 \times 2 \times 2$ է, կատարելով ստորև ներկայացված գործողությունը, որը շատ նման է մեկ շերտանի նկարի դեպքին, կստանանք $2 \times 2 \times 1$ չափանի զանգված, որտեղ վերջին չափողականությունը ֆիլտրերի քանակն է ցույց տալիս։



Օրինակ եթե ուզում ենք ոչ միայն ուղղահայաց գծերը գտնենք, այլ նաև հորիզոնական, պետք է կիրառենք երկու ֆիլտրեր և կունենանք $2 \times 2 \times 2$ չափի զանգված։ Այս օրինակում ունենք երեք շերտ այդ պատճառով զանգվածի չափսը $3 \times 3 \times 2$ է։



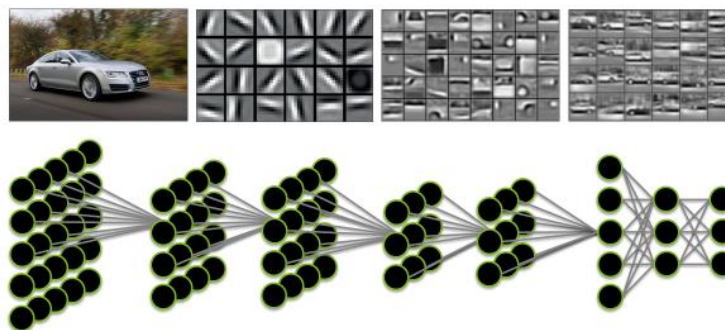
Ընդհանուր դեպքում եթե ունենք $n_h \times n_w \times c_0$ չափի նկար (որտեղ c_0 -ն նկարի խորությունն է), կիրառելով c_1 հատ ֆիլտրեր, որոնցից յուրաքանչյուրը $k_h \times k_w \times c_0$ չափանի է, և վերցնելով (p_h, p_w) չափի padding և (s_h, s_w) չափի stride, կունենանք հետևյալ չափի զանգված։

$$\frac{(n_h + 2p_h - k_h + 1)}{s_h} \times \frac{(n_w + 2p_w - k_w + 1)}{s_w} \times c_1$$

Մեքենայական ուսուցման զարգացման զուգընթաց, մարդիկ հասկացան, որ ֆիլտրներ հնարելու փոխարեն կարելի են մեքենային սովորեցնել, որպեսզի նա ինքը այնպիսի ֆիլտրեր օգտագործի, որը հնարավորինս լավ կտարբերակի օբյեկտի դետալները և հենց ինքը օբյեկտին: Դրա համար ֆիլտրերի արժեքները վերցնում ենք որպես պարամետրեր և ամեն քայլի ժամանակ փորձում ենք փոխել այդ պարամետրերը այնպես, որ լավագույնս տարբերակի օբյեկտները միմյանցից:

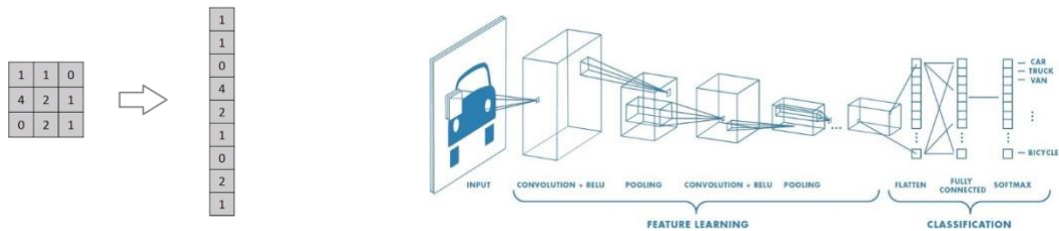
Որպեսզի ստանանք CNN-ի առաջին շերտը, նկարի վրա convolution operation ենք կիրառում յուրաքանչյուր ֆիլտրով և գումարում ենք որոշակի շեղում յուրաքանչյուր ֆիլտրից ստացված արդյունքի վրա: Այդ ստացված արդյունքները տեղադրելով ակտիվացնող ֆունկցիայի մեջ, ստանում ենք CNN-ի առաջին շերտը: Եթե կիրառել ենք c_1 հատ ֆիլտրեր, ապա առաջին շերտում կունենանք c_1 դետալ տարբերակող նեյրոններ, որոնք պատասխանատու են փոքր դետալների համար օրինակ՝ գծեր, կորեր և այլ փոքր դետալներ: Այնուհետև այս c_1 խորությամբ զանգվածի վրա կիրառելով նույն գործողությունները տարբեր պարամետրերով կստանանք CNN-ի նոր շերտ, որը պատասխանատու կլինի ավելի բարդ դետալներ տարբերակելու համար: Օրինակ ավտոմեքենայի նկար ճանաչելու դեպքում անիվ կամ հայելի:

HOW A DEEP NEURAL NETWORK SEES



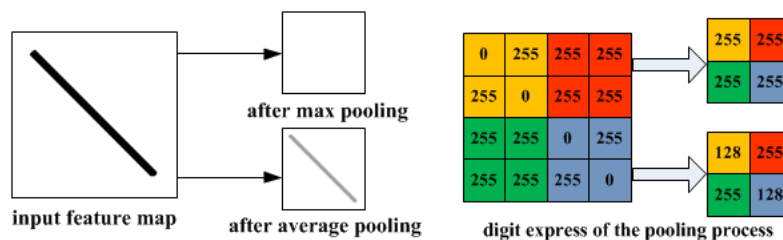
Դասավորելով ստացված թվերը մեկ տողում և կատարելով MLP ցանցի

գործողությունները կտարբերակենք օրինակ ավտոմեքենաների տեսակները միմյանցից:

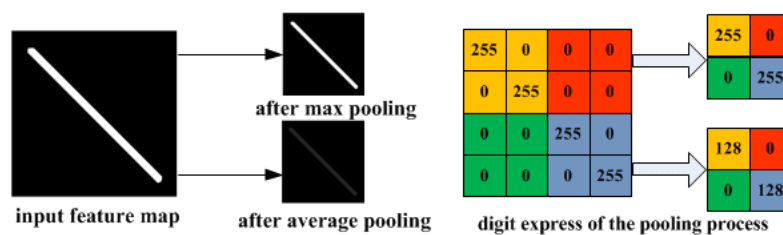


8.6 Pooling layer

Հաճախ CNN-ի մեջ կիրառում են pooling operation-ը որը շատ նման է convolution operation-ին, բայց այս դեպքում ֆիլտրերի արժեքները չենք բազմապատկում նկարի արժեքներով այլ վերցնում ենք, կա'ն այդ տարածքի միջինը (average pooling), կա'ն մաքսիմումը (max pooling): Սրա նպատակը հիմնականում հաշվարկները քչացնելն է, և սրա միջոցով ավելի է ընգծվում դետալները և որոշակիորեն լուծվում է overfitting-ի խնդիրը:



(a) Illustration of max pooling drawback

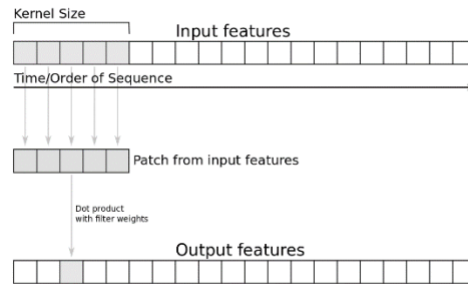


(b) Illustration of average pooling drawback

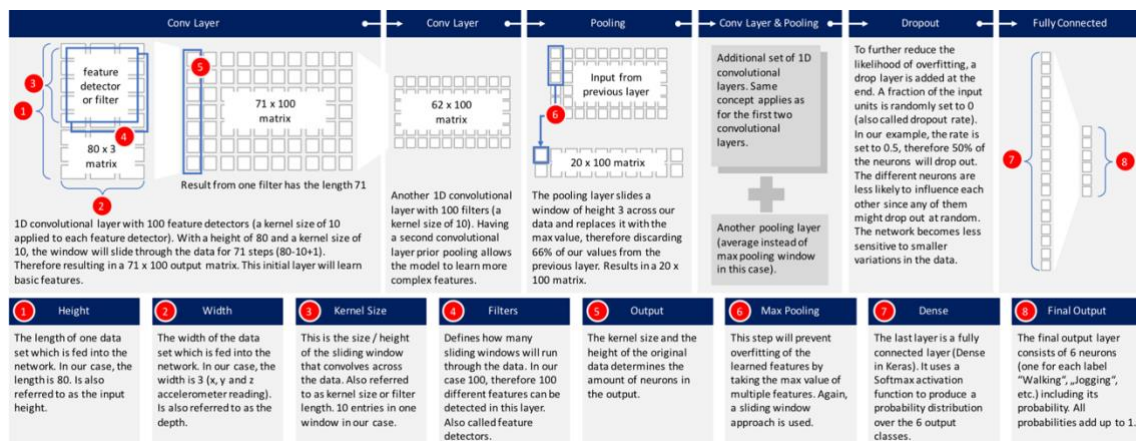
CNN-ը Ժամանակային շարքերի համար

Վերոնշյալ CNN մոդելներ նկարներ տարբերակելու համար էին, բայց հնարավոր է CNN-ը օգտագործել ժամանակային շարքերի համար ևս: Ժամանակային շարքերը իտարբերություն նկարների ունեն ընդհամենը մեկ չափողականություն, դրա համար որոշակի ձևափոխություններ է պետք

կատարել որպեսզի այն օգտագործվի նաև ժամանակային շարքերի համար: Մենք ժամանակային շարքը կարող ենք բաժանել տարբեր մասերի և դրանք համարել որպես նկար որոնք ունեն մեկ պիքսել երկարություն և լայնությունը հավասար է բաժանված մասի երկարությանը: Այս պարագայում ֆիլտրերը ևս ունեն մեկ չափողականություն: ԵՎ կատարելով convolution operation-ը ինչպես երկչափ CNN-ի պարագայում կստանանք նոր շարք:

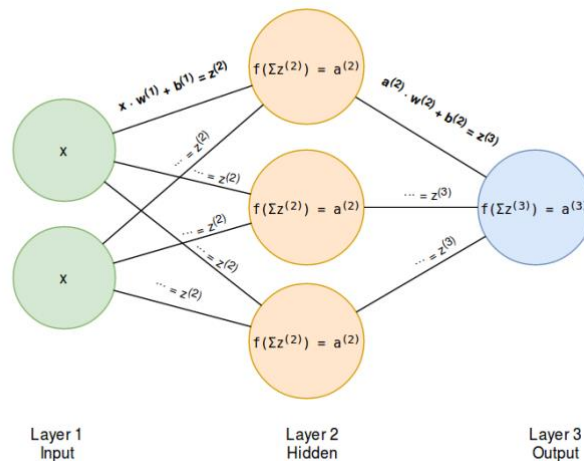


Եթե ցանկանում ենք վերլուծել միքանի ժամանակային շարքեր միառժամանակ, այդ պարագայում մենք ուղղակի կարող ենք համարել որ ժամանակային շարքը կազմված է օրինակ 3 «գույն»-ից: ԵՎ նույն տրամաբանության շարունակելով և օգտագործելով տարբեր ֆիլտրեր կստանանք միքանի շարքեր, որոնցից ամեն մեկը պատասխանատու է որևէ երևույթ բացահայտելու համար: Այնուհետև դրանք դասավորելով մեկ տողում և կիրառելով MLP ցանցի քայլերը՝ կտարբերակենք ժամանակային շարքերը միմյանցից:

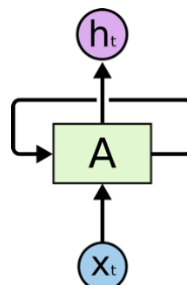


9. Ռեկուրենտ նեյրոնային ցանցեր

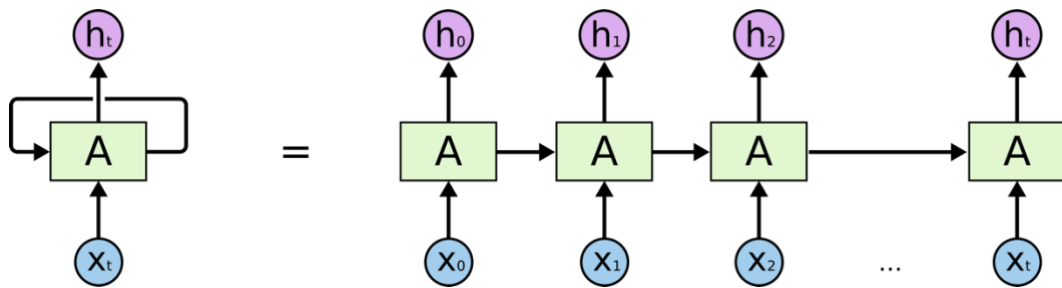
MLP նեյրոնային ցանցերը ունեն հետևյալ տեսքը: Նրանք ամեն անգամ որպես input վերցնում են վեկտորներ, և կատարելով որոշակի գործողություններ, վերադարձնում են որևէ արդյունք: Այսինքն ամեն անգամ մեր input տվյալները միմյանցից անկախ են:



Այս մոդելը հարմար չէ ժամանակային շարքերը կանխատեսելու համար: Օրինակ՝ բաժնետոմսի այսօրվա գինը կախված է երեկվա գնից և այսօրվա որևէ իրադարձությունից: Այս պարագայում օգնության են գալիս Ռեկուրենտ նեյրոնային ցանցերը (այսուհետև՝ RNN): RNN-ի պարագայում նեյրոնային ցանցերը ունեն հետևյալ տեսքը:

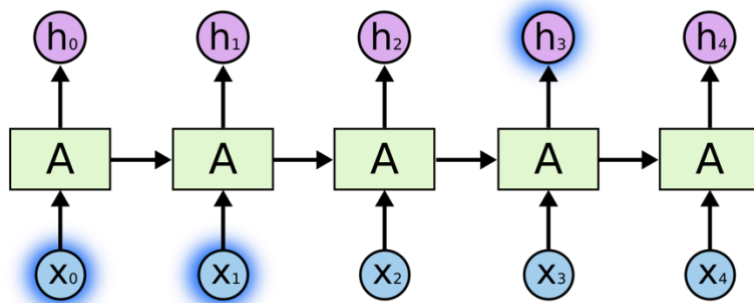


Ինչպես տեսնում ենք այն իր մեջ ներառում է ցիկլ: Ամեն անգամ նոր արդյունք վերադարձնելիս այն հաշվի է առնում նախորդ արդյունքը: Ավելի պարզ տեսնելու համար ներկայացնենք բացված տեսքով :



Ինչպես տեսնում ենք առաջին քայլում վերցնելով x_0 մեզ վերադարձնում է h_0 : Երկրորդ քայլում հաշվի առնելով նախորդ քայլում կանխատեսված h_0 -ն և նոր x_1 վեկտորը, կանխատեսում է երկրորդ ցուցանիշը:

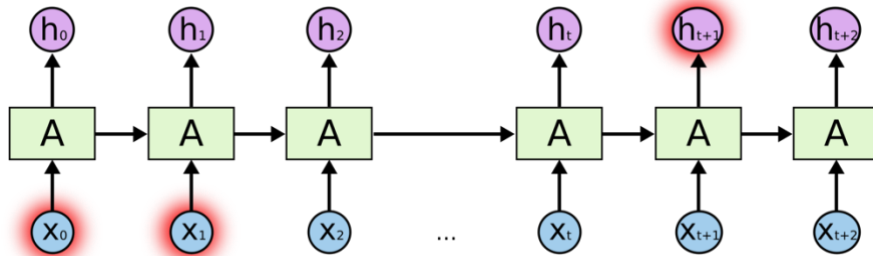
Այս ցանցը բավականին լավ է աշխատում կարճաժամկետ կանխատեսումներ կատարելու համար:



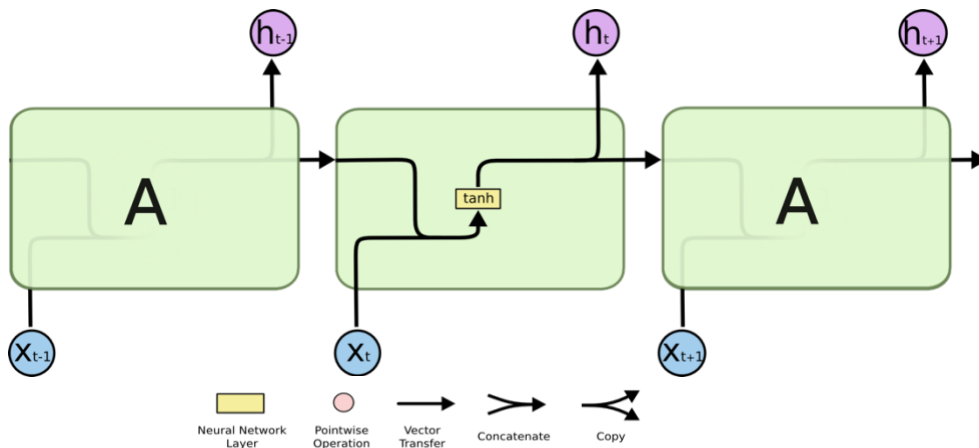
Օրինակ եթե ունենք նախադասություն «Արևը գտնվում է....» և ցանկանում ենք կանխատեսել հաջորդ բառը ինչ կլինի: Կիրառելով RNN՝ այն կվերադարձնի միգուցե «երկնքում» բառը: Այս նախադասությունը ավարտելու համար RNN-ին որևէ այլ նախադասությունից կողմնակի ինֆորմացիա հարկավոր չէ:

Բայց լինում են դեպքեր, որ նախադասությունը ավարտելու համար պետք է նախկինից իմանալ որոշակի ինֆորմացիա:

Օրինակ՝ «Ես ծնվել եմ Հայաստանում: Իմ մայրենի լեզուն», այս պարագայում երբ շատ ենք հեռանում Հայաստան բառից, ցավոք RNN-ը ուղակի մոռանում տեքստի բովանդակությունը անհետացող գրադիենտի պատճառով:



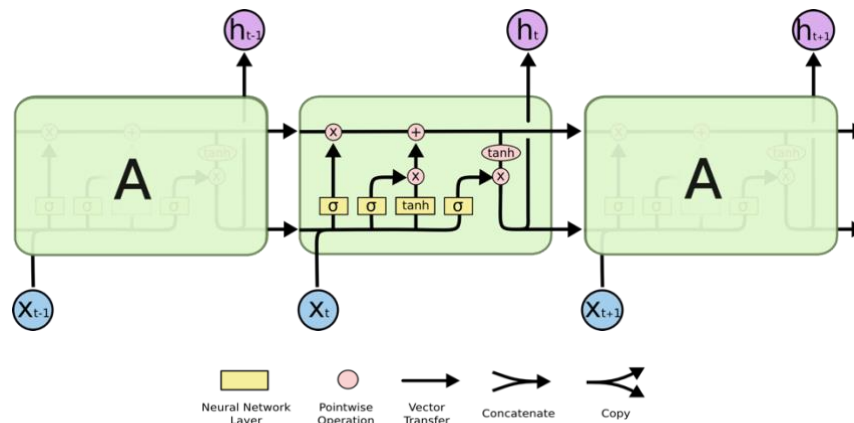
Գրեթե բոլոր ռեկուրենտ նեյրոնային ցանցերը կարող ենք ներկայացնել կրկնվող նեյրոնային ցանցերի շղթայի միջոցով: RNN-ի պարագայում այն ունի հետևյալ տեսքը:



9.1 Long Short Term Memory networks

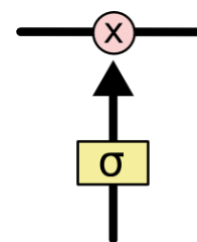
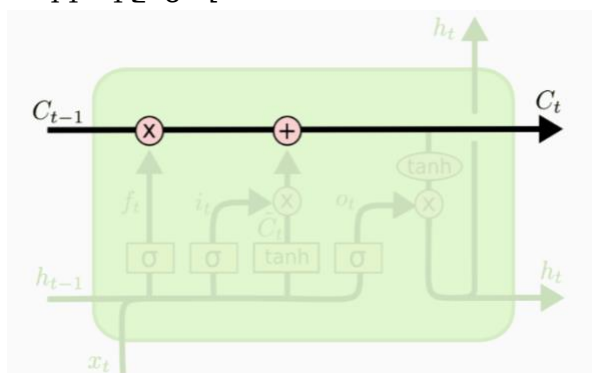
Անհետացող գրադիենտից խուսափելու մեթոդներից մեկը Long Short Term Memory networks (LSTM) ալգորիթմն է: Այս ալգորիթմը իր մեջ պահում է երկար ժամանակ առաջ եղած ինֆորմացիան:

LSTM-ի պարագայում շղթան ունի հետևյալ տեսքը:



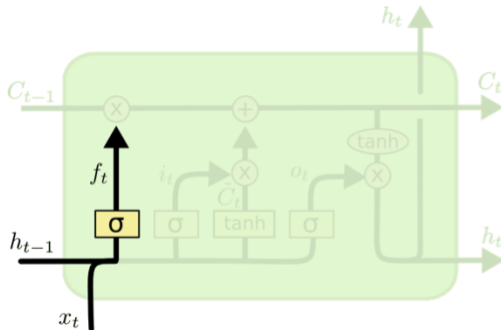
Ինչպես տեսնում ենք այն բավականաչափ հետաքրքիր է կառուցված: Այժմ դիտարկենք թե ինչ է ներառվում մոդուլի մեջ:

Մոդուլի մեջ գտնվող հորիզոնական գիծը կոչվում է բջջի վիճակ: LSTM-ի հիմնական գաղափարը սրա մեջ է: LSTM-ի ունի հնարավորություն հեշտորեն ավելացնելու կամ պակասեցնելու ինֆորմացիա բջջի վիճակի մեջ դարպասների միջոցով:



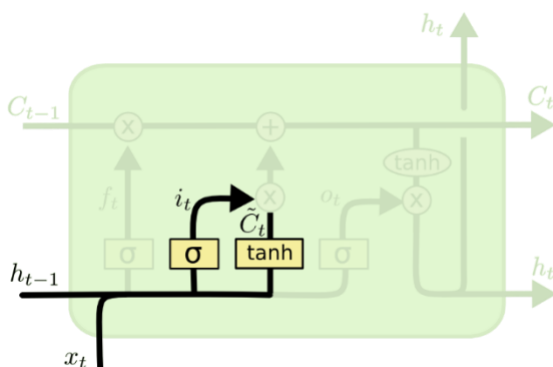
Առաջին քայլով LSTM-ը որոշում է, թե որ ինֆորմացիան պետք չէ ներառվի բջջի վիճակի մեջ: Այն կատարվում է սիգմոյիդ շերտի միջոցով որը կոչվում է “forget gate layer”: Այն միավորելով x_t -ն և h_{t-1} -ն, տալով նրանց կշիռներ և գումարելով որոշակի շեղում, տեղադրում է սիգմոյիդ ֆունկցիայի մեջ: Այն վերադարձնում է 0-ի և 1-ի միջև գտնվող վեկտոր: 1-ը նշանակում է ամբողջապես պահիր ինֆորմացիան, իսկ 0-ն ոչինչ մի պահիր: Օրինակ՝

Արամը լավ մարդ է: Բայց Գոռը չար է: Երբ ալգորիթմը տեսնում է վերջակետը և Գոռ բառը, նա մոռանում է նախորդ նախադասության բովանդակությունը և սա կատարվում է forget gate-ի միջոցով:



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Հաջորդ քայլում որոշում է ինչ նոր ինֆորմացիա պահի բջջի վիճակի մեջ: Սա բաղկացած է երկու մասից: Առաջին քայլով սիգմոյիդ ֆունկցիայի միջոցով որոշվում է, որ արժեքները պետք է պահվեն բջջի վիճակի մեջ: Երկրորդ քայլում ստեղծել վեկտոր, որը պարունակում այն բոլոր հնարավոր արժեքներ որոնք կարող են ավելացվել բջջի վիճակի մեջ: Սա կատարվում է tanh ֆունկցիայի միջոցով, որը ընդունում է արժեքներ -1 ից 1 միջակայքում: Բազմապատկում ենք այս երկու արդյունքները և ստանում ենք կարևոր ինֆորմացիա, որը պետք է ավելացվի բջջի վիճակին:



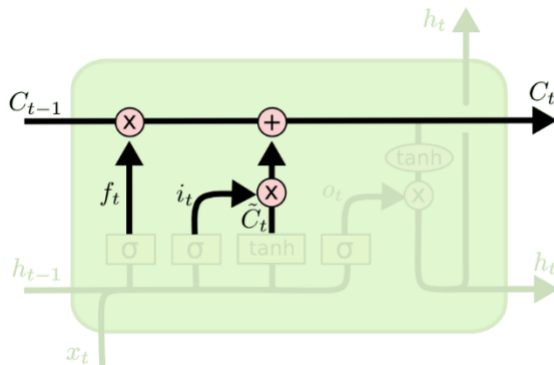
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Օրինակ՝ Արամը լողալ գիտի: Նա ինձ հեռախոսով ասաց, որ նա նավաստի է: Այստեղ կարևոր ինֆորմացիա է օրինակ, որ Արամը լողալ գիտի և նավաստի է, բայց հեռախոսով ասելը այդքան էլ կարևոր չէ և կարող է անտեսվել:

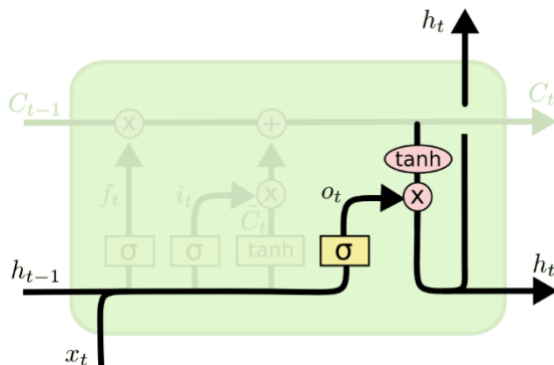
Հաջորդ քայլում պետք է C_{t-1} -ից անցնենք նոր C_t բջջի վիճակին: Բազմապատկելով նախորդ վիճակը f_t -ով ալգորիթմը մոռանում է այն, ինչ

նախորդք որոշել էինք որ պետք է մոռանալ: Այնուհետև գումարելով նախորդ արդյունքին ստանում ենք նոր բջջի վիճակ:



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

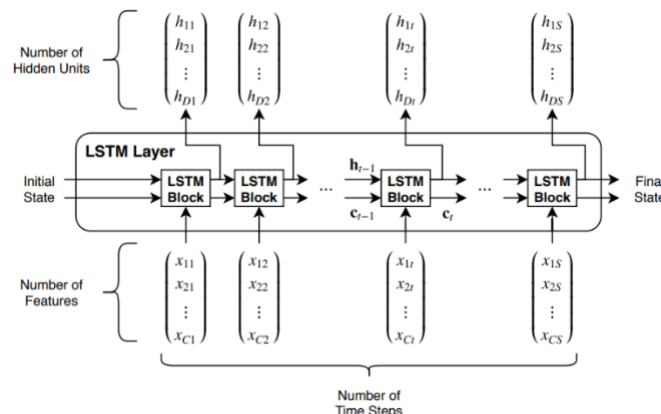
Վերջին քայլում որոշում ենք output-ը: Սկզբում որոշում ենք բջջի վիճակի որ մասն ենք ներկայացնելու որպես output: Դա անում են սիգմոյիդ ֆունկցիայի միջոցով: Այնուհետև վիճակի վեկտորի վրա կիրառում ենք tanh ֆունկցիան որպեսզի ստանանք -1-ից 1 միջակայքում գտնվող թվեր: Բազմապատկելով նախորդ քայլում ստացված արդյունքի հետ, արտածում ենք այն մասը, որը որոշել էինք արտածել:



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Ժամանակային շարքերի համար LSTM-ը ունի հետևյալ տեսքը: Այն ընդունելով մի քանի հաջորդական տվյալներ՝ կանխատեսում է հաջորդ ժամանակի արժեքները:



10. Տվյալների նախապատրաստում

Ներդրնային ցանցեր կառուցելու համար անհրաժեշտ են տվյալների որոշակի կառուցվածք: Նախնական տվյալները չէին բավարարում այդ կառուցվածքին: Անհրաժեշտ էր տվյալները տողերում դասավորել ըստ ժամանակահատվածի, և ամեն ժամանակահատվածի համար ունենալ ask և bid order-ները և դրանց համապատասխանող քանակները դասավորված ըստ level-ների:

Time					
1:00 pm	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_4(t)$	$x_5(t)$
1:10 pm	$x_1(t+1)$	$x_2(t+1)$	$x_3(t+1)$	$x_4(t+1)$	$x_5(t+1)$
1:20 pm	$x_1(t+2)$	$x_2(t+2)$	$x_3(t+2)$	$x_4(t+2)$	$x_5(t+2)$
1:30 pm	$x_1(t+3)$	$x_2(t+3)$	$x_3(t+3)$	$x_4(t+3)$	$x_5(t+3)$
...					
2:30 pm	$x_1(t+9)$	$x_2(t+9)$	$x_3(t+9)$	$x_4(t+9)$	$x_5(t+9)$

Դասավորելով այդպես ստացվեց, որ LOB-ը ունի 200-ից ավել level և քանի որ բարձր level-ներում գտնվող տվյալները այքան էլ ազդեցություն չունեն բաժնետոմսի գնի վրա, որոշվեց օգտագործել level 10 LOB տվյալներ, հետազոտության համար: Այն ժամանակահատվածները որոնց համար միայն կային ask և bid order-ներ, հեռացվեցին տվյալների միջից: Այսպես ստացվեց որ երկու իրար հաջորդող պահերի հեռավորությունը միջինում 22 վայրկյան է: Կային ժամանակահատվածներ որոնք ունեին օրինակ միայն level 2 տվյալներ, հաջորդ level-ների համար դրանք լցվեցին իրենց նախորդող level-ի թվով: Ամեն պահի միջին գինը հաշվել է level 1 ask-ի և level 1 bid-ի գները միջինացնելով:

10.1 Տվյալների պիտակավորում

Որպեսզի լուծենք խնդիրը նախևառաջ պետք է ունենանք պիտակավորված տվյալներ: Տվյալները պիտակավորելու համար պետք է սահմանենք թե ինչ ենք հասկանում բաժնետոմսի գնի աճ կամ նվազում ասելով: Կան մի քանի պիտակավորման մեթոդներ, որոնցից ունանք փորձել եմ տվյալների վրա:

Սահմանենք $m_-(t)$ և $m_+(t)$ մեծությունները:

$$m_-(t) = \frac{1}{k} \sum_{i=1}^k p_{t-i}$$

$$m_+(t) = \frac{1}{k+1} \sum_{i=0}^k p_{t+i}$$

Որտեղ t -ն ցույց է տալիս t -րդ պահին է, k -ն՝ ժամանակային հորիզոնը և p_t -ն t -րդ պահի գինը:

Պիտակավորման մեթոդ 1

t -րդ պահի համար սահմանենք l_t մեծությունը հետևյալ կերպ

$$l_t = \frac{m_+(t) - p_t}{p_t}$$

Եթե $l_t > \alpha$, ապա t -րդ պահը կպիտակավորենք որպես գնի աճ, եթե $l_t < -\alpha$ ապա որպես նվազում, մնացած դեպքերում կհամարենք որ գինը չի փոխվել: Այստեղ α -ն նախորոք որոշված հաստատուն է:

Պիտակավորման մեթոդ 2

$$l_t = \frac{m_+(t) - m_-(t)}{m_-(t)}$$

Եթե $l_t > \alpha$, ապա t -րդ պահը կպիտակավորենք որպես գնի աճ, եթե $l_t < -\alpha$ ապա որպես նվազում, մնացած դեպքերում կհամարենք որ գինը չի փոխվել:

Պիտակավորման մեթոդ 3

$$l_t = \frac{p_t - m_-(t)}{m_-(t)}$$

Եթե $l_t > \alpha$, ապա t -րդ պահը կպիտակավորենք որպես գնի աճ, եթե $l_t < -\alpha$ ապա որպես նվազում, մնացած դեպքերում կհամարենք որ գինը չի փոխվել:

Պիտակավորման մեթոդ 4

$$l_t = \frac{p_{t+k} - p_t}{p_t}$$

Եթե $l_t > \alpha$, ապա t -րդ պահը կպիտակավորենք որպես գնի աճ, եթե $l_t < -\alpha$ ապա որպես նվազում, մնացած դեպքերում կհամարենք որ գինը չի փոխվել:

Վերջնական մոդելներ կառուցվել են 4-րդ մեթոդով պիտակավորված տվյալների վրա:

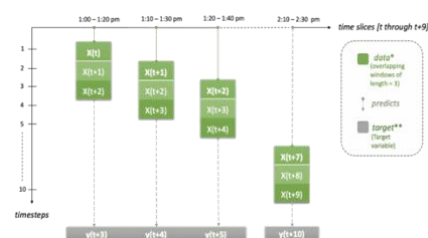
Time						
1:00 pm	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_4(t)$	$x_5(t)$	$y(t)$
1:10 pm	$x_1(t+1)$	$x_2(t+1)$	$x_3(t+1)$	$x_4(t+1)$	$x_5(t+1)$	$y(t+1)$
1:20 pm	$x_1(t+2)$	$x_2(t+2)$	$x_3(t+2)$	$x_4(t+2)$	$x_5(t+2)$	$y(t+2)$
1:30 pm	$x_1(t+3)$	$x_2(t+3)$	$x_3(t+3)$	$x_4(t+3)$	$x_5(t+3)$	$y(t+3)$
...						
2:30 pm	$x_1(t+9)$	$x_2(t+9)$	$x_3(t+9)$	$x_4(t+9)$	$x_5(t+9)$	$y(t+9)$

Տվյալները պիտակավորելուց հետո տվյալները անհրաժեշտ էր բաժանել այնպիսի մասերի (պատուհանների), որը անհրաժեշտ է CNN և LSTM մոդելներ կառուցելու համար: Դրա համար պետք է սահմանել 3 փոփոխական՝

- Այն տվյալների քանակը որոնց հիման վրա պետք է կանխատեսել ապագա գնի պիտակը: Սա պայմանականորեն անվանենք տվյալների պատուհանի մեծություն:
- Ժամանակային հորիզոն:
- Քանի քայլով պետք է տեղաշարժել տվյալների պատուհանը:

$$X = \begin{bmatrix} x(t) \\ x(t+1) \\ \vdots \\ x(t+n-1) \end{bmatrix} = \begin{bmatrix} x_1(t) & x_2(t) & x_3(t) & x_4(t) & x_5(t) & x_6(t) \\ x_1(t+1) & x_2(t+1) & x_3(t+1) & x_4(t+1) & x_5(t+1) & x_6(t+1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(t+n-1) & x_2(t+n-1) & x_3(t+n-1) & x_4(t+n-1) & x_5(t+n-1) & x_6(t+n-1) \end{bmatrix}$$

$$y = \begin{bmatrix} y(t) & y(t+1) & y(t+2) & y(t+3) & \dots & y(t+n) \end{bmatrix}$$



Հետազոտության ժամանակ պատուհանի մեծությունը վերցվել է 50, ժամանակային հորիզոնը 5 և քայլերի քանակը 1: Այսինքն նախորդ 50 տվյալների օգտագործելով փորձվել է կանխատեսել բաժնետոմսի գնի ուղղությունը 5 պահ հետո: Եթե այս թվերը վերածենք ժամանակի, կստացվի որ միջինում նախորդ 18 րոպեների order-ների հիման վրա փորձում ենք կանխատեսել թե մոտավորապես 2 րոպե հետո բաժնետոմսի գինը կաճի, կնվազի թե՞ կմնա նույնը:

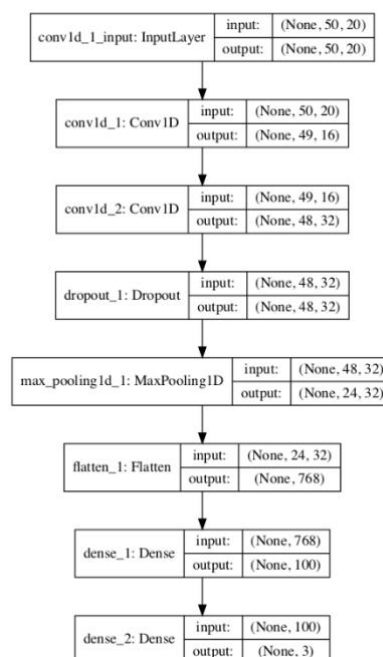
Հետազոտության ընթացքում կառուցվել են 3 մոդել, որոնք կառուցվածքը և արդյունքները ներկայացված են հաջորդ գլուխներում:

11. CNN մոդելի արդյունքը

Ինչու՞ CNN

CNN-ը հնարավորություն է տալիս գտնել թաքնված օրինաչափություններ ժամանակային շարքերում, որոնք նախկինում հնարավոր չէր լինում գտնել.

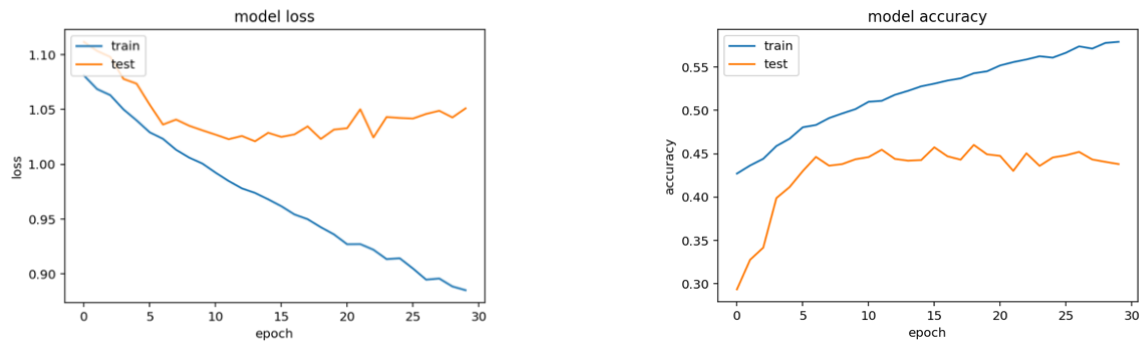
Փորձելով CNN-ի տարբեր կառուցվածքներ՝ լավագույն արդյունքը ստացվեց հետևյալ կառուցվածքի պարագայում:



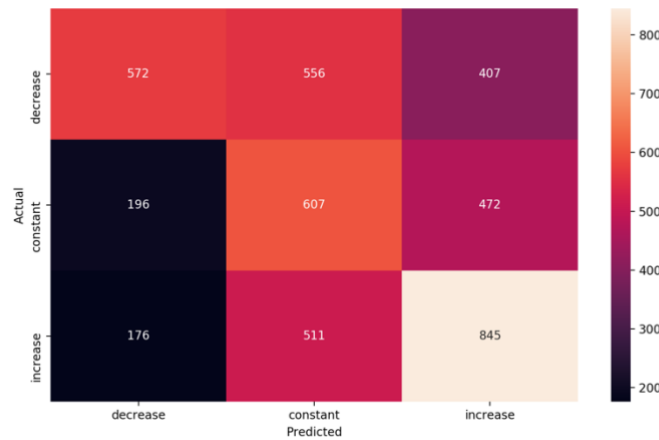
Ցանցի առաջին շերտում գտնվում են 16 ֆիլտրներ: Երկրորդ շերտում 32 ֆիլտրներ: Այնուհետև dropout տեխնիկայի միջոցով անջատվել են ցանցի 0.4 տոկոսը և կիրառվել է maxpooling overfitting-ից խուսափելու համար:

Այնուհետև հարթեցնելով արդյունքը հաջորդ շերտերում կիրառվել է 100 նեյրոն ունեցող MLP և softmax շերտ, որպեսզի ստանանք դասերին պատկանելու հավանականությունները:

Օպտիմիզացման ալգորիթմն ընտրվել է Adam ալգորիթմը և որպես loss function վերցվել է categorical crossentropy ֆունկցիան: Ուսուցման ընթացքում կիրառվել են learning rate decay և early stopping տեխնիկաները: Ուսուցման գործընթացը ունի այս տեսքը:



Այս մոդելը ճիշտ կանխատեսեց test set-ի տվյալների 46%-ի պիտակները: Confusion matrix-ը, precision և recall-ը և F1-score-երը հետևյալն են:



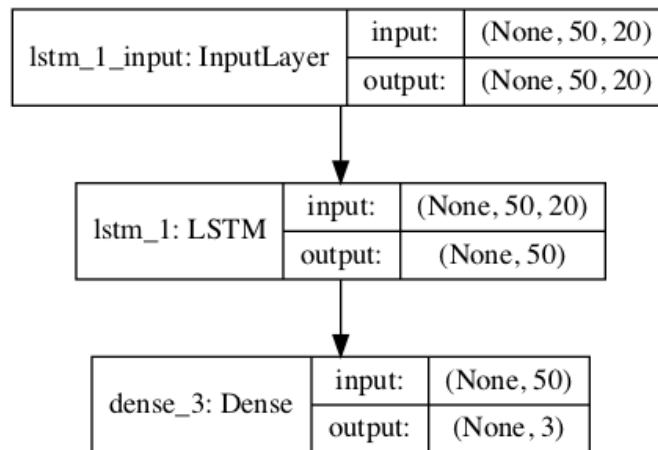
	precision	recall	f1-score
decrease	0.61	0.37	0.46
constant	0.36	0.48	0.41
increase	0.49	0.55	0.52

12. LSTM մոդելի արդյունքը

Ինչու՞ LSTM

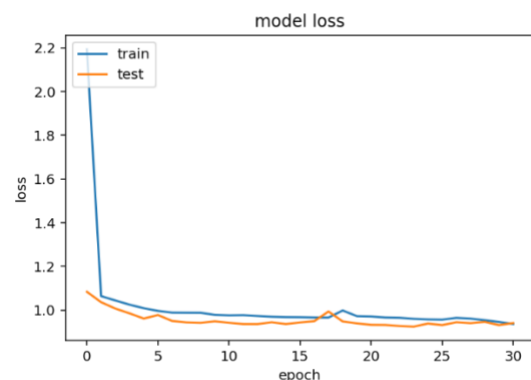
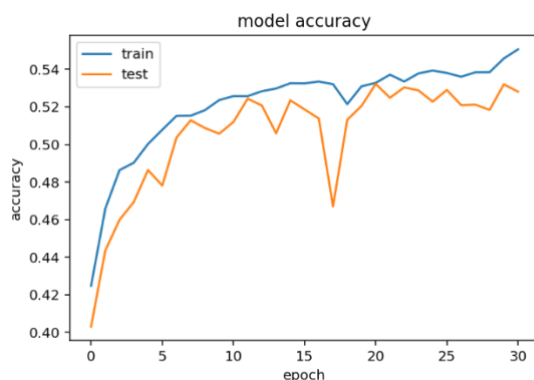
LSTM-ը հնարավորություն է տալիս բացահայտել երկարաժամկետ կապեր երկու ժամանակների միջև:

LSTM-ի պարագայում լավագույն արդյունքը ստացվեց հետևյալ կառուցվածքի պարագայում:

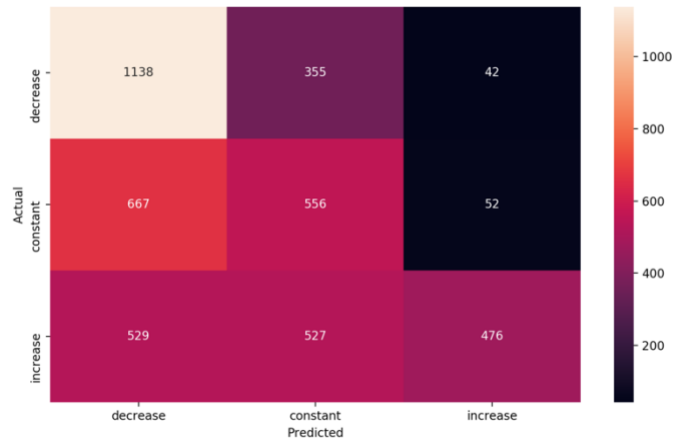


Առաջին շերտում LSTM-ի 50 նեյրոն և հաջորդ շերտում softmax շերտ:

LSTM-ի պարագայում ևս կիրառվել են Adam ալգորիթմը և categorical crossentropy loss function-ը: Ուսուցման գործընթացը ունի այս տեսքը:



LSTM-ի պարագայում կանխատեսելու ճշգրտությունը ստացվեց 50%:
Confusion matrix-ը, precision և recall-ը և F1-score-երը հետևյալն են:



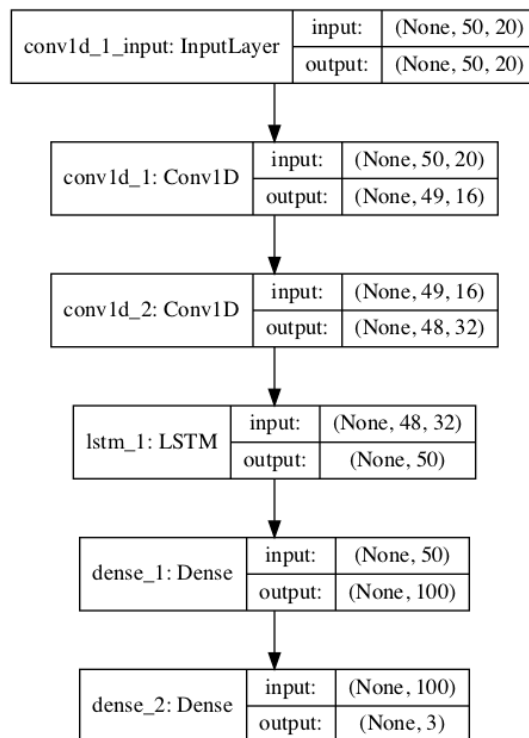
	precision	recall	f1-score
decrease	0.49	0.74	0.59
constant	0.39	0.44	0.41
increase	0.84	0.31	0.45

13. CNN-LSTM մոդել արդյունքը

Ինչու՞ CNN-LSTM

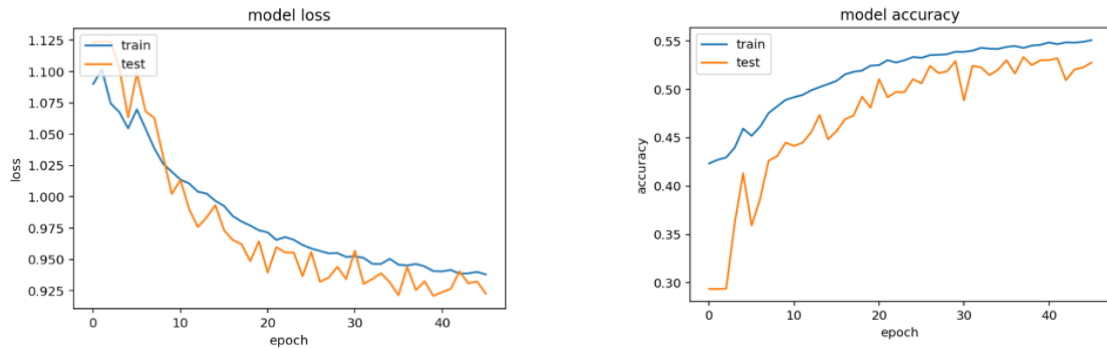
Այս մոդելը հնարավորություն է տալիս բացահայտելու և՛ թաքնված երևույթները, և՛ երկու ժամանակների միջև կապերը:

Համադրելով վերոնշյալ երկու մոդելները կառուցվեց ցանց, որը ունի հետևյալ կառուցվածքը:

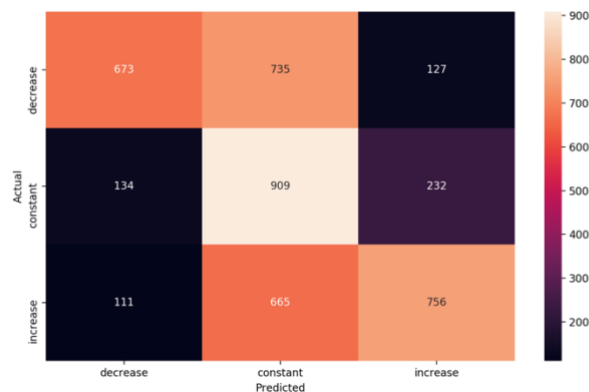


Առաջին երկու շերտերում համապատասխանաբար CNN-ի երկու շերտ՝ 16 և 32 ֆիլտրերով: Այնուհետև LSTM-ի 50 նեյրոն և MLP-ի 100 նեյրոն և վերջում softmax շերտ:

Այս մոդելի ընթացքում կիրառվել են ինչպես Adam այնպես էլ stochastic gradient descent օպտիմալիզացման ալգորիթմները: Ուսուցման գործընթացը հետևյալ տեսքն ունի:



Այս մոդելի արդյունքում, test set-ի ճշգրտությունը հասավ մինչև 0.538%-ի: Confusion matrix-ը, precision և recall-ը և F1-score-երը հետևյալն են:



	precision	recall	f1-score
decrease	0.73	0.44	0.55
constant	0.39	0.71	0.51
increase	0.68	0.49	0.57

14. Համեմատություն այլ մոդելների հետ

Համացանցում սակավաթիվ աշխատանքներ կան, որոնք փորձում են կանխատեսել բաժնետոմսի գնի փոփոխությունը LOB տվյալների հիման վրա՝ օգտագործելով վերոնշյալ մոդելները [1],[2] : Այս աշխատանքում կառուցված ցանցերը ավելի պարզ և փոքր են և զբաղեցնում են ավելի փոքր ծավալ, որոնք հնարավորություն են տալիս ավելի արագ կանխատեսելու բաժնետոմսի գնի ուղղությունը: Ստորև ներկայացված են այս աշխատանքի և [1] աշխատանքի արդյունքները:

Model	Mean Precision	Mean Recall	Mean F1-Score
CNN	0.49	0.47	0.46
LSTM	0.44	0.59	0.50
CNN-LSTM	0.60	0.54	0.54

Աղյուսակ 1 Այս մոդելը

Model	Mean Precision	Mean Recall	Mean F1-Score
CNN	0.44	0.54	0.43
LSTM	0.45	0.55	0.42
CNN-LSTM	0.45	0.56	0.44

Աղյուսակ 2 Այլ մոդել

Ինչպես տեսնում ենք արդյունքները բավականին մոտ են ստացվել: Համեմատելու համար՝ այս աշխատանքի ցուցանիշներից հանենք [1]-ի ցուցանիշները:

Model	Precision Diff	Recall Diff	F1-Score Diff
CNN	0.05	-0.07	0.03
LSTM	-0.01	0.04	0.08
CNN-LSTM	0.15	-0.02	0.10

Աղյուսակ 3 Մոդելների տարբերությունը

Եզրակացություն

Այսպիսով կիրառելով պիտակավորման 4-րդ մեթոդը LOB տվյալների համար և կառուցելով CNN, LSTM և CNN-LSTM մոդելներ այդ տվյալների վրա, հնարավոր եղավ կանխատեսել բաժնետոմսի գնի ուղղությունը հաջորդ երկու րոպեների ընթացքում համապատասխանաբար 46%, 49%, 54% տոկոս ճշգրտությամբ, որը համեմատելի է միջև այժմ եղած մոդելների հետ:

Կառուցված մոդելների կշիռները ունեն փոքր ծավալ, ինչը հնարավորություն է տալիս ցանկացած ցուցանիշ կանխատեսել միլիվայրկյանների ընթացքում, որը խիստ կարևոր է այն մարդկանց համար որոնք զբաղվում են ինտենսիվ առևտրով: Այս մոդելները նաև կարող են օգտագործվել ավտոմատ առևտուր կատարող ալգորիթմներում որպեսզի հաղորդեն լրացուցիչ կարճաժամկետ ինֆորմացիա և կարող են հանդիսանալ որպես ուղղենիշներ ընկերությունների համար՝ զերծ պահելու բաժնետոմսի գինը անցանկալի տատանումներից:

Հետագայում հնարավոր է բարելավել մոդելների որակը, ցանցերի կառուցվածքը փոխելով, այլ պիտակավորման մեթոդներ կիրառելով և ավելի շատ տվյալներ օգտագործելով:

Գրականության ցանկ

Հոդվածներ

1. [“Using Deep Learning for price prediction by exploiting stationary limit order book features”](#); Avraam Tsantekidisa, Nikolaos Passalisa, Anastasios Tefasa, Juho Kannianenb, Moncef Gabboujc, Alexandros Iosifidis; 2018 year
2. [“DeepLOB: Deep Convolutional Neural Networks for Limit Order Books”](#); Zihao Zhang, Stefan Zohren, and Stephen Roberts; 2020 year
3. [“Human activity recognition based on time series analysis using U-Net”](#); Yong Zhang, Yu Zhang, Zhao Zhang, Jie Bao, Yunpeng Song; China
4. [“1D Convolutional Neural Networks and Applications – A Survey”](#); Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, Daniel J. Inman
5. [“U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging”](#); Mathias Perslev, Michael Hejlselbak Jensen, Sune Darkner, Poul Jørgen Jennum, Christian Igel; 2019 year
6. [“Long-term Recurrent Convolutional Networks for Visual Recognition and Description”](#); Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell; 2016 year
7. [“Efficient Dense Labeling of Human Activity Sequences from Wearables using Fully Convolutional Networks”](#); Rui Yao, Guosheng Lin, Qinfeng Shi , Damith Ranasinghe; 2017 year
8. [“Deep learning for time series classification: a review”](#); Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller; 2019 year
9. [“Deep Learning Specialization”](#); Andrew Ng; 2020 year
10. [“Get Started with Using CNN+LSTM for Forecasting”](#); Yitong Ren; 2019
11. [“Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences”](#); Nils Ackermann; 2018 year
12. [“How to Develop Convolutional Neural Network Models for Time Series Forecasting”](#); Jason Brownlee; 2019 year

13. [“How to Develop 1D Convolutional Neural Network Models for Human Activity Recognition; Jason Brownlee”; 2019 year](#)
14. [“Deep Learning for Time Series Classification: a brief overview”; Hassan Ismail Fawaz; 2019 year](#)
15. [“Human Activity Recognition \(HAR\) Tutorial with Keras and Core ML \(Part 1\)”; Nils Ackermann; 2018 year](#)
16. [“Human Activity Recognition using LSTMs on Android — TensorFlow for Hackers” \(Part VI\); Venelin Valkov; 2017 year](#)
17. [“Gentle Dive into Math Behind Convolutional Neural Networks”; Piotr Skalski; 2019 year](#)
18. [“A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”; Sumit Saha; 2018 year](#)
19. [“The Problem of Vanishing Gradients”, Animesh Agarwal; 2019 year](#)
20. [“How to Develop Multi-Step LSTM Time Series Forecasting Models for Power Usage”; Jason Brownlee; 2018 year](#)
21. [“How to Choose Loss Functions When Training Deep Learning Neural Networks”; Jason Brownlee; 2020 year](#)
22. [“A Beginner's Guide to LSTMs and Recurrent Neural Networks”; Søren Kierkegaard](#)
23. [“Essentials of Deep Learning: Introduction to Long Short Term Memory”; Pranjali Srivastava; 2017 year](#)
24. <https://www.youtube.com/watch?v=WCUNPb-5EYI>
25. <https://www.youtube.com/watch?v=y7qrilE-Zlc>
26. <http://dprogrammer.org/rnn-lstm-gru>
27. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
28. <https://github.com/deKeijzer/Multivariate-time-series-models-in-Keras>
29. https://github.com/miroblog/limit_orderbook_prediction

App

1. [“Dive into Deep Learning”; Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola; 2019 year](#)
2. “Predict the Future with MLPs, CNNs and LSTMs in Python”; Jason Brownlee; 2018 year
3. [“Logistic Regression”; Daniel Jurafsky & James H. Martin; 2019 year](#)
4. [“Human Activity Recognition Based on Motion Sensor Using U-Net”; Yong Zhang, Zhao Zhang, Yu Zhang, Jie Bao, Yifan Zhang, Haiqin Deng; 2019 year](#)

Հավելված

Այստեղ կցում եմ github-ի հասցեն՝ որտեղ գտնվում են բոլոր աշխատանքային
ֆայլերը: <https://bit.ly/2XsjGAx>