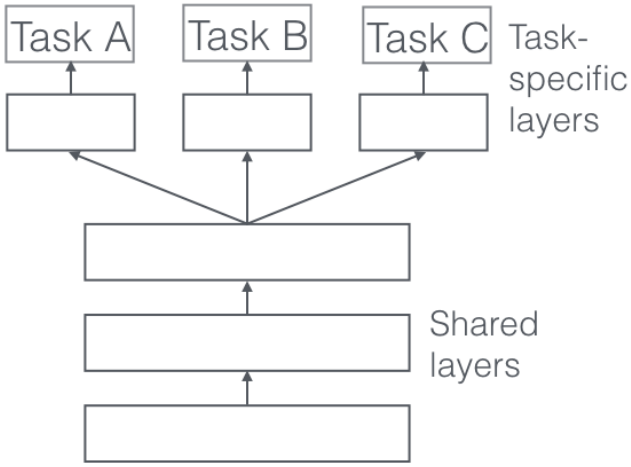
Բազմախնդիր ուսուցում

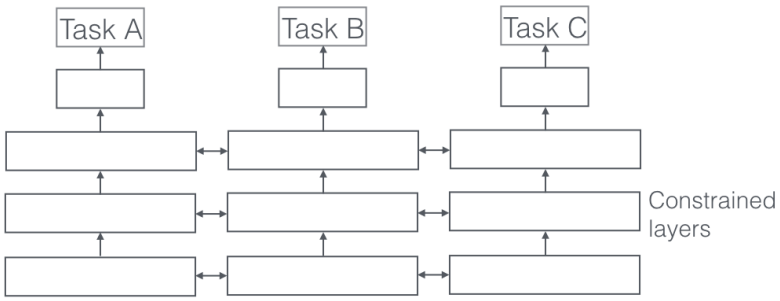
Հայկ Կարապետյան

Մինչ այս նեյրոնային ցանցերով մենք լուծում էինք միայն մեկ խնդիր (classification կամ regression կամ segmentation)։ Հիմա կծանոթանանք, թե ինչպես կարելի է մի ցանցի օգնությամբ լուծել միանգամից երկու կամ ավելի խնդիրներ։ Inception ցանցին ծանոթանալիս, տեսանք որ ցանցում կարող է լինեկ մեկից ավելի կորստի ֆունկցիա։ Դա օգնում էր խուսափել vanishing gradient խնդրից և ցանցի ներքևի շերտերը ավելի լավ էին սովորում ընդհանուր հատկանիշներ (եզրեր առանձնացնել)։ Մի քանի խնդիր մեկ ցանցի օգնությամբ լուծելու տարբերակներից մեկը նույն մեթոդի կիրառումն է։ Ցանցը մի քանի մասի է բաժանվելու և դրվեն տարբեր կորստի ֆունկցիաներ։ Օրինակ՝ պետք է կատարենք նկարի դասակարգում մարդ կա, թե ոչ և միևնույն ժամանակ կատարենք մարդու segmentation։ Կարող ենք սկզբում ունենալ 3 շերտ և վերջին շերտից ճյուղավորվի երկու մաս։ Առաջին ճյուղավորման կորուստը կլինի մարդու լինել կամ չլինելը, իսկ երկրորդինը segmentation-ը (Նկար 1)։



Նկար 1։ Ներքևի շերտերը երկեք խնդրի համար էլ ընդհանուր են, իսկ վերևի մասում կիսվում են ըստ խնդիրների

Ներքևի շերտերի քանակը ինչքան ուզենք կարող ենք փոփոխել և նույն ձևով վերևի շերտերը։ Task A-ն կարող է ունենալ 5 շերտ, Task B-ն 6 շերտ, Task C-ն 4 շերտ։ Վերին շերտերը high-level feature-ներն են, որոնք անհրաժեշտ են ցանցին վերջնական որոշում կայացնելու համար և բոլոր ցանցերում տարբեր են, իսկ ներքևինները low level feature-ներն են և նման խնդիրներում կարող են շատ նման լինել կամ այս դեպքում նույնը լինելը։ Դրանք սովորում են ընդհանուր feature-ներ (օրինակ՝ եզրագծեր առանձնացնել)։
Երկրորդ եղանակը, առանձին low level feature-ներ ունենալն է, բայց այնպես անելով, որպեսզի երկու խնդիրների low level feature-ները շատ չտարբերվեն (Նկար 2)։



Նկար 2։ Ներքևի շերտերը երեք խնդրի համար էլ նման են։ Ներքևի շերտերի քանակը բոլորի համար նույնն է

Հիմա դիտարկենք, թե ինչպես կարող ենք մի քանի շերտեր իրար նման պահել: Իրար նման շերտեր նշանակում է, որ նման են այդ շերտերի կշիռները: Regularization անցնելիս մենք սահմանափակում էինք կշիռները, որպեսզի շատ չմեծանան և դա կատարում էինք կորստի ֆունկցիայի միջոցով՝ $L = L_{pred} + \lambda \sum_{i=1}^{k} (w_k)^2$: λ-ն այստեղ հիպերպարամետր է և ընդունում է 1-ից փոքր արժեք, որպեսզի մեր հիմնական կորստին (իրական պիտակի և գուշակած պիտակի) ավելի մեծ ուշադրություն դարձնենք: Նույն կերպ այստեղ կարող ենք օգտագործել regularization և նմանեցնել շերտերի կշիռները:

$$L = L_{pred} + \lambda \sum_{i=1}^{k} (w_{Ai} - w_{Bi})^2 + (w_{Bi} - w_{Ci})^2$$

λ-ն նորից հիպերպարամետր է և դրա արժեքը փոքր է 1-ից: Այս կերպով նույնպես կարող ենք միաժամանակ մի քանի խնդիր ուսուցանել ցանցին և դրա ներքևի շերտերը իրար նման կլինեն:
Հիմնականում մեզ անհրաժեշտ չի լինում մի մոդելի միջոցով լուծել երկու խնդիր, բայց multitask learning կարող ենք կիրառել նաև մեկ մոդել ուսուցանելիս: Segmentation-ի տվյալներ գտնելը բավականին բարդ է, քանի որ նկարի միջից մարդուն եզրագծերով առանձնացնելը բավականին ժամանակ է պահանջում: Այդ պատճառով առկա տվյալները քիչ են: Այս դեպքում մենք կարող ենք ցանցի ներքևի շերտերը ուժեղացնել ավելի շատ տվյալների վրա ուսուցանելով, իսկ վերևի մասը ավելի քիչ տվյալների վրա: Օրինակ՝ ուզում ենք ստանալ segmentation-ի մոդել: Կարող ենք ցանցը կիսել և մի մասում ավելացնել մարդու classification-ի կորստի ֆունկցիա, իսկ մյուս մասում segmentation-ի կորստի ֆունկցիա: Այն նկարները, որոնք ունեն և՛ segmentation պիտակ, և՛ classification պիտակ կարող ենք կորուստը հաշվել հետևյալ բանաձևով.

$$L = L_{segmentation} + \lambda L_{classification}$$

Իսկ այն նկարները որոնք կունենան միայն classification պիտակը՝ մարդ կա կամ չկա, կորստի ֆունկցիան կլինի՝ $L = \lambda L_{classifcation}$: λ-ն նորից հիպերպարամետր է և 1-ից փոքր արժեք է, քանի որ մեր հիմնական խնդիրը մարդու segmentation-ն է: Classification-ի տվյալները կօգնեն ներքևի շերտերին (low level feature) ավելի լավ ուսուցանվել: Այսինքն multitask learning կարող ենք կիրառել հետևյալ դեպքերում.

1. Ցանկանում ենք ուսուցանել մեծ ցանց, որ միաժամանակ կկարողանա լուծել մի քանի խնդիր
2. Ունենք մի քանի խնդիր և նման կամ նույն low level feature-ներ ունենալը կաող է դրանց համար օգտակար լինել:
3. Մի խնդրի համար ունենք քիչ տվյալներ, իսկ մյուսի համար ավելի շատ տվյալներ և երկու խնդրի համար էլ low level feature-ները նման են: