Математичка гимназија

Матурски рад

-из предмета Програмирање и програмски језици-

Увод у напредујуће неуралне мреже

Аутор
Николија Бојковић

Ментор
Проф. Петар Радовановић

Садржај

1 Увод

1.1 Увод у машинско учење

Моћ рачунара из дана у дан расте. Њиховом применом се обавља широк спектар задатака у различитим сферама. Међутим, увек има места за напредак и циљ је да рачунари могу обавити још многе функције. Алан Тјуринг је током Другог светског рата помагајући Алијанси у декодирању Енигме, поставио важно питање *Могу ли машине да мисле?* Био је фасциниран интелигенцијом и размишљањем, те је и осмислио тест 1950. године, познат под називом *Тјурингов тест*, који је увод у развој вештачке интелигенције. Његово питање је било да ли машина може да размишља паметно као и човек.

Идеја је да се игра игра у којој учествују две особе и једна машина. Машина и једна особа су изоловани у различитим просторијама. Задатак друге особе је да погоди у којој просторији је машина тако што поставља питања на која они одговарају. Уколико особа која пита не може погодити где је машина, тада машину сматрамо интелигентном. Такав интелигентан уређај и даље не постоји. Међутим, та идеја представља основ развоја вештачке интелигенције. Примена тога се огледа и у академске, али и индустријске сврхе. Током 21. века условљен је невероватан напредак индустрије, јер је постало могуће користити и радити са великим базама података, као и применити машиснко учење. Ова област се активно развија од педесетих година прошлог века, то јест од конструисања перцептрона, први систем који учи једноставне законитости.

Циљ машинског учења јесте да програм даје тачно решење и на примерима које није видео. Тако се у банкама ова метода користи за откривање лажних трансакција, емаил сервиси користе за филтритање спам поште итд. Такође, примена машинског учења је видна у медицинским истраживањима и здравству, где се огледа у бржем и лакшем дијагностиковању. Помоћу ове методе постало је могуће идентификовати стадијум болести. Машинско учење се најчешће дели на три категорије, на основу од података којима се служи. Дате категорије су:

1. **Супервизирано учење** (енг. Supervised learning) – рачунару су дати улазни подаци, као и излазни подаци за сваки од датих уноса. Задатак алгоритма јесте да пронађе правило које повезује улазе са излазима и на примерима који нису задани раније.

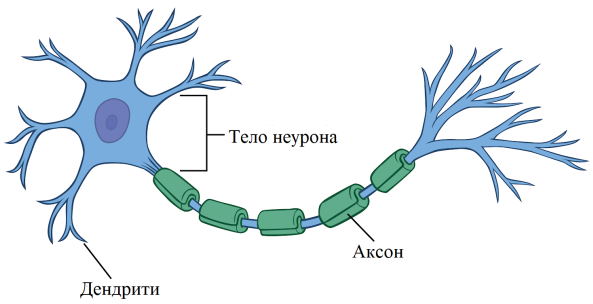
Ови алгоритми се најешће користе код проблема класификације, на пример у препознавању да ли је тумор малигдан или бенигдни.

2. **Несупервизирано учење** (енг. Unsupervised learning)– рачунару су дати само улазни подаци. Задатак рачунара јесте да нађе везу међу њима и да их сам групише према сопственој процени и генерише решење.

3. **Учење са појачањем** (енг. Reinforcement learning) – Рачунар је у константној интеракцији са динамичким окружењем у коме треба да изврши задатак (нпр. Аутоматска вожња аутомобила) где након сваке одлуке добија повртану информацију у виду награде и казне у зависности од одлуке које је донео.

2 Увод у неуралне мреже

Неуралне мреже су област машинског учења и вештачке интелигенције подстакнута и инспирисана принципима који се односе на функционисање учења људи. Из тог разлога је дошло до идеје о коришћењу методе познате као неуралне мреже. Моделоване су по узору на наш нервни систем мозга. Предност коришћења вештачких неуралних мрежа се огледа у могућности моделирања комплексних проблема без потребе за експлицитним дефинисањем математичког модела. Ова област је почела да се развија средином 20. века. Прва тренирана неуралн мрежа била је демонстрирана 1957. године од стране Franka Rosenblatta.



Слика 1: Неурон

2.1 Начин функционисања неуралне мреже

Неурон има интересантну структуру. Састоји се од дендрита који примају сигнал, тела неурона који обрађује сигнал и аксона који га преноси. Веза између свака два неурона је остварена синапсом, то јест простором између дендрита и аксона два суседна неурона.

Попут нервног система који се састоји од мноштва повезаних неурона, тако се и неуралне мрежа састоји од чворова, вештачких неурона, који су међусобно повезани. Грубо објашњени ток рада вештачког неурона:

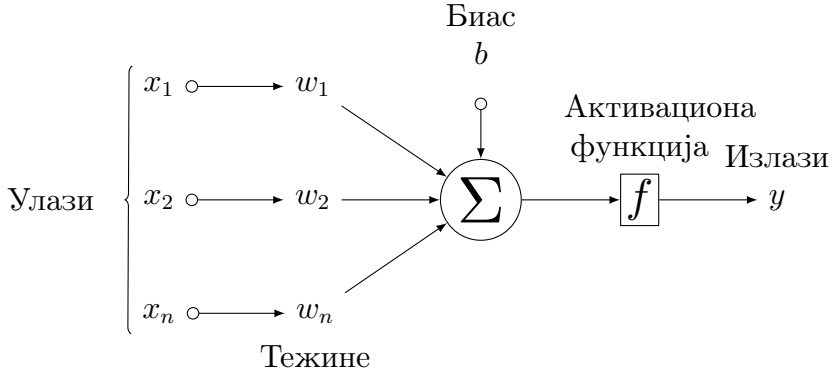
- Прима сигнале од суседних чворова
- Извршава одређене рачунице
- Шаље сигнал суседним чворовима

Аналогија између неурона и вештачког неурона	
Неурон	Вештачки неурон
Језгро	Чвор
Дендрит	Улаз
Синапса	Тежине
Аксон	Излаз

Генерално, неуралне мреже могу да уче кроз стварање нових конекција, брисање постојећих конекција, мењање тежина, мењање граничне вредности функције перцептрона (о томе ће бити више речи касније), стварање нових неурона, брисање већ постојећих неурона итд.

2.2 Перцептрон

Перцептрон представља јединицу грађе нашег математичког модела неуралне мреже. Вештачки неурони су повезани тако да између свака два суседна постоји синапса. Свакој синапси је додељена бројна вредност, односно тежина. Тежине имају пресудну улогу у тренирању модела помоћу машинског учења.



Приказ рада перцептрона

Имамо n улаза, где је i-том улазу (x_i) придружена тежина (w_i). Након примања сигнала рачуна се скаларни производ улазних података и тежина коме се додаје добијамо:

$$\sum_{i=1}^{n} x_i w_i + b$$

На дату вредност рачунице се примењује активациона функција и добијамо излаз. Биас вредност омогућава да контролишемо излаз за нула-улаз.

Најједноставнији пример рада перцепрона је случај када имамо само један улаз, u. Тада важи:

$$\text{Излаз перцептрона: } f(u) = \begin{cases} 0, & \text{ако } u < C \\ 1, & \text{ако } u >= C \end{cases}$$

Приказан је график дате функције. Слично, у случају да имамо

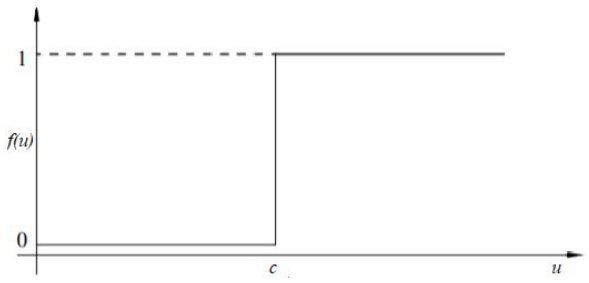


График функције $f(u)$

више улаза важи следећа релација:

$$\textbf{Излаз} = \begin{cases} 0, & \text{ако } \sum w_i x_i < C \\ 1, & \text{ако } \sum w_i x_i \geq C \end{cases} \text{ где је } C \text{ дата гранична вредност.}$$

То је класичан математички модел перцептрона. Како би нам појам перцептрона био интуитивно јаснији можемо да размишљамо о њему као о уређају који доноси одлуке одмеравањем доказа. На пример, претпоставимо да ће се за викенд у твом граду одржати концерт бенда који волиш слушати. Ти желиш да одлучиш да ли да идеш на концерт. Донећеш одлуку на основу следећа три фактора:

1. Да ли је време топло без кише?

2. Да ли твоји пријатељи желе да ти се придруже?

3. Да ли имаш превоз до места одржавања концерта?

Представимо ова три фактора као следеће променљиве редом x_1, x_2 и x_3. Уколико је први фактор задовољен $x_1 = 1$, у супротном $x_1 = 0$. Аналогно дефинишемо вредности за факторе x_2 и x_3.

Претпостави да је то концерт твог омиљеног бенда тако да си заинтересован да идеш иако твоји пријатељи не буду могли. Такође, желиш да идеш и у случају да је тешко стићи до локације одржавања. Међутим, теби је важно какво је време. У случају да је кишовито, ти нећеш ићи. Можемо користити перцептрон за доношење ове одлуке. Једна од могућности јесте да тежине буду $w_1 = 7$ за време, $w_2 = 2$ и $w_3 = 2$ за остала два фактора. Како је w_1 највеће вредности то индукује да ти време највише значи, много више него друштво пријатеља и превоз. Нека је гранична вредност C једнака 5. Тада перцептрон враћа вредност 1 у сваком случају када је време лепо, а 0 увек када није лепо. Такав модел не прави разлику у томе да ли ће други и трећи фактор бити задовољени.

Примећујемо да варирањем граничних вредности добијамо различите резултате. Дефинишимо биас b, тако да буде исте апсолутне вредности, а различитог знака од граничне вредности C. Одатле следи:

$$\textbf{Излаз} = \begin{cases} 0, & \text{ако } \sum w_i x_i + b < 0 \\ 1, & \text{ако } \sum w_i x_i + b \geq 0 \end{cases}$$

Пример:
Претпоставимо да имамо две улазне промењиве x и y за неки перцептрон P. Нека су A и B тежине за x и y редом. Како перцептрон враћа не-нула вредност уколико скаларни производ улазних података премашује граничну вредност C.
Излаз перцептрона:

$$P = \begin{cases} 0, & \text{ако } Ax + By < C \\ 1, & \text{ако } Ax + By \geq C \end{cases}$$

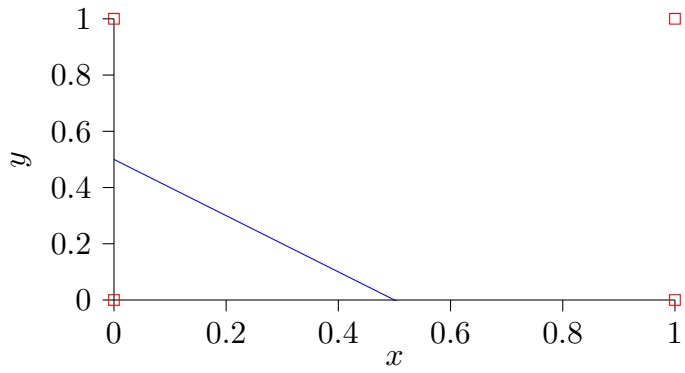
Сетимо се да су $Ax + By \geq C$ и $Ax + By < C$ два простора xy равни подељена правом $Ax + By + C = 0$.

Дата је таблица која приказује логичку функцију коњукцију ($\wedge$):

1	0	1
0	0	0
$\wedge$	0	1

Нека је $A = 1$, $B = 1$ и $C = 0.5 \implies x + y = 0.5$

Приказан је график где означеним тачкама y координата означава вредност извршене функције $x \wedge y$. На датом графику је црвеном бојом приказана та права и уочавамо да тачке $(0, 1), (1, 0), (1, 1)$ и $(0, 0)$ не припадају истом делу равниограниченом функцијом и координантним осама. На сличан начин функционише процес класификације.



2.3 Познате активационе функције

Активациона функција се бира у зависности од проблема који треба бити решен. Могу бити сложене. Неке од најпознатијих су:

- Сигмоидна функција : $f(x) = \frac{1}{(1+e^x)}$
- Хиперболички тангенс:$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- ReLu: $f(x) = max(0, x)$
- Leaky ReLu: $f(x) = max(\alpha x, x)$

2.3.1 Сигмоидна функција

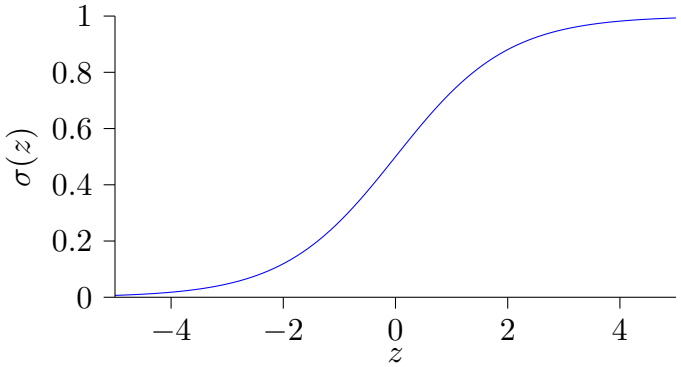
Претпоставимо да имамо мрежу повезаних перцептрона чији је задатак да реши неки проблем. На пример, улази су матрице пиксела од скениране слике ручно написане цифре. Задатак мреже је да научи тежине и биасе како би могла класификовати цифре исправно. Међутим, мале промене у тежини или биасу могу довести до грешке. Зато се врло често користи Сигмоидна функција, која гласи:

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

Сигмоидни неурони су слични перцептронима. Исто као и перцептрон улази су x_1,x_2... Међутим, перцептрон као излаз показује вредности 0 или 1, док ова функција узима вредности из интервала $[0, 1]$.

Како бисмо разумели сличност између перцептрона и сигмоидног неурона претпоставимо да $z \equiv wx + b$ где $z \to \infty$. Тада $\lim_{z\to\ \infty} \sigma(z) = 1$ што исто важи и за перцептрон. За $z \to -\infty$ важи $\lim_{z\to\ -\infty} \sigma(z) = 0$ што исто важи и за перцептрон.

Сигмоидна функција



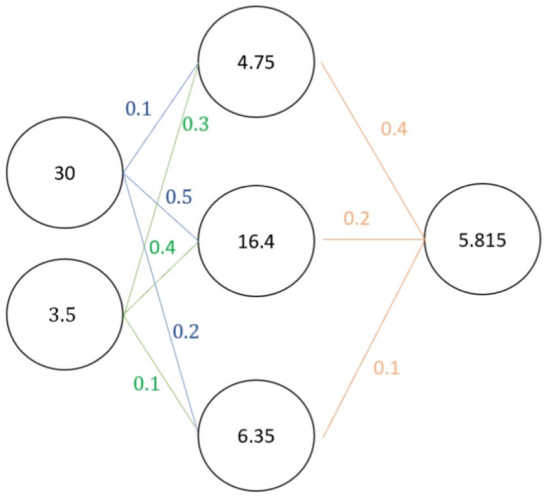
2.4 Пример

Посматрајмо податке у табели.

Број сати учења	Просек	Резултат теста
30	3.5	положио
12	2	пао
10	2.2	пао
34	4	положио

Тада имамо улаз приказан са две карактеристике (x_1 = број сати учења, x_2 = просек и y = резултат теста). Представљена је неуралнa мрежа првог улаза на слици.

Дате тежине су произвољно одабране (касније се обично врши опти-

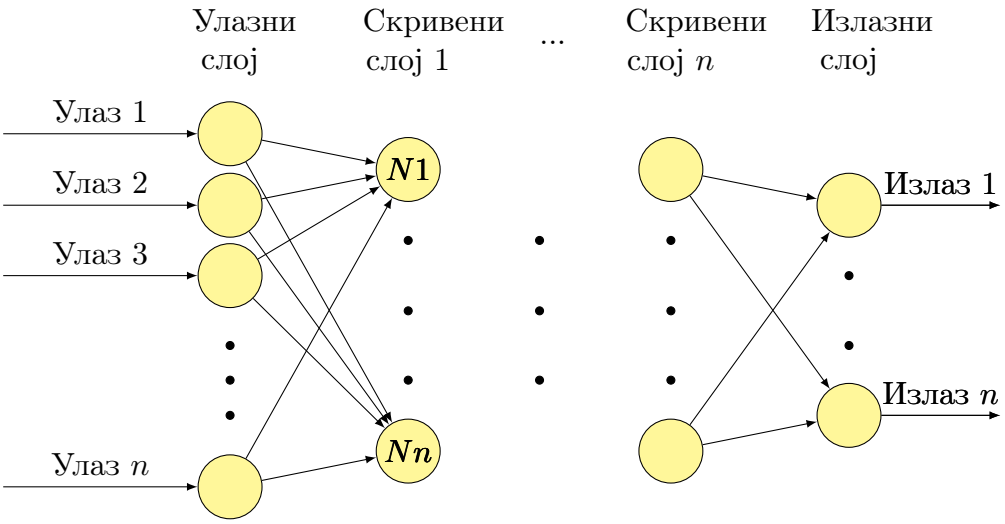


Пример неуралне мреже првог улаза

мизација, како би функција губитка била што мања (о томе ће бити речи касније), али је увек прво неопходно иницијализовати, а касније следи оптимизација). У нашем случају добијамо да је излаз једанак 5.815. Нека је Сигмоидна функција активациона. Како она константну вредност слика у опсегу $[0, 1]$, закључујемо да је то вероватноћа да је неко положио испит. На пример, уколико је излаз 0.75 то значи да је вероватноћа да је та особа положила испит једнака 75%. У нашем случају вредност Сигмоидне функције је 0.997, што значи да је вероватноћа да је испит положен 99.7%.

2.5 Грађа неуралне мреже

Неуралне мреже се обично састоје из три слоја улазни, скривени и излазни. Улазни слој прима податке из спољашње средине, скривени прослеђује и обрађује релевантне податке до излазног слоја. Излазни слој представља слој чији је излаз коначан резултат рада наше мреже.



Структура неуралне мреже

Неуралне мрежа је састављена од више међусобно повезаних перцептрона. Можемо је изградити на следеће начине:

- **Напредујуће** (ацикличне, енг. feedforward) - Добијена мрежа (граф) нема циклусе, односно излаз из једног перцептрона иде у други и не враћа се у тај исти. Односно састоји се из више слојева где сваки слој сем излазног шаље податке у наредни и сваки перцпетрон тог слоја прима одређене податке.
- **Рекурентно** - Добијена мрежа (граф) има циклусе

2.5.1 Почетна конфигурација multylayer мреже

Број слојева
Најмањи број слојева јесте два (улазни и излазни слој). Међутим, број скривених слојева варира. Идеја је да се крене од једног скривеног слоја и да се тај број повећава све док се не дође до адекватног броја којим би наш проблем био решив.

Број неурона
Број неурона улазног и излазног слоја је дефинисан. Међутим, за скривене слојеве не постоји стандардан број. Најчешће се крене са малим бројем неурона који се тренирањем повећава.

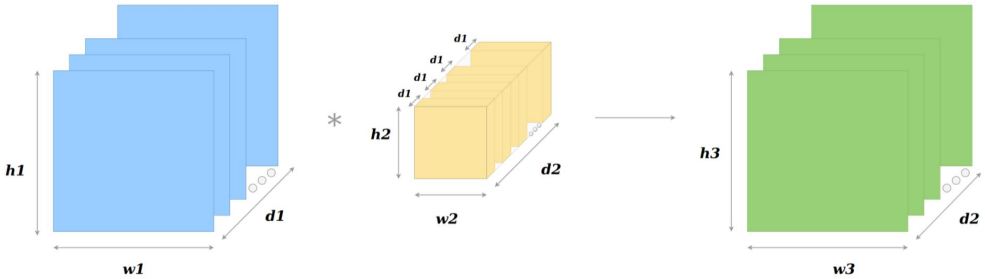
Одабир активационе функције
Опет и одабир активационе функције зависи од формулације проблема. Активациона функција може бити другачија за различите слојеве. У највећем броју случајева је активациона функција иста за све скривене и излазни слој.

3 Конволуцијска неуронална мрежа

Конволуцијска неуронална мрежа се најчешће користи за анализу слике и сигнала. То је напредујућа неуронална мрежа. Састоји се из три слоја:

1. **Улазни слој**
2. **Скривени слој**:
   (а) конволуцијски слој
   (б) слој сажимања
   (ц) потпуно повезани слој
3. **Излаз мреже**

Како бисмо појаснили рад скривених слојева конволуцијске неуралне мреже посматраћемо пример када су улази слике. Слика је ништа друго до матрица вредности пиксела. Представља се својом дужином, ширином и вредностиума пиксела. Уколико је слика у боји, сваки од три канала (RGB) се обично представља вредности пиксела у опсегу 0-255. RGB слика се представља улазном структуром коју називамо улазна запремина (енг. input volume).



Кернел

У обради слике кернел (конволуцијска матрица) служи за замућење,

0	2
3	-1

Пример 2×2 кернела

изоштравање, откривање ивица итд. Врло често се назива и филтер. Кернел представља дводимензионалну матрицу малих димензија у односу на слику на коју се примењује и састоји се од реалних вредности. Примена кернела се назива конволуција. У зависности од тога који филтер употребимо добићемо другачију слику.

Карактеристика

Оно што је такође важно јесте битна карактеристика (енг. feature) слике. Врло често се можемо срести израз користан образац (енг. useful pattern). Карактеристику добијамо из слике коју у мрежу доводимо путем улазног слоја. Конволуцијска мрежа учи карактеристике улазних слика којима се неуралне мрежа тренира.

Рецептивно поље

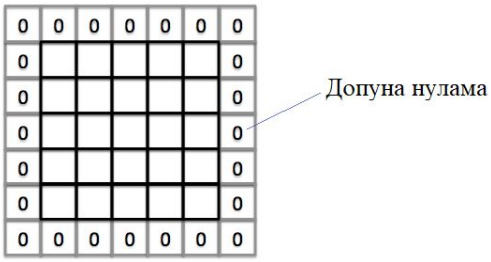
Није практично да сваки неурон једног слоја буде повезан са свим неуронима суседног слоја, јер може довести до великог броја тежина. Из тог разлога су неурони распоређени у 3D структуру где се сваки неурон повезује само са одређеним неуронима наредног слоја. Рецептивно поље представља део слике које је видљив филтеру у датом тренутку.

Корак

Корак се односи на величину за коју померамо наш филтер.

Допуна

Најчешће се врши допуна нулама, односно одрежени редови/колоне се

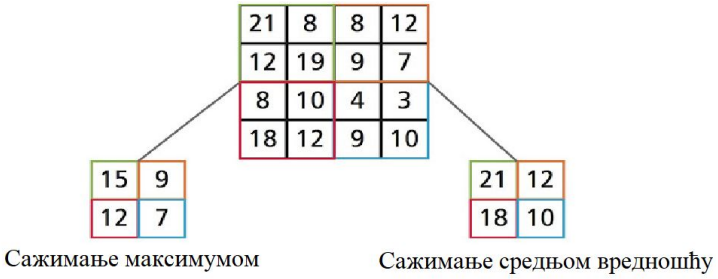


Допуна нулама

допуњују нулама како бисмо применили филтер на њу (јер смањењем димензија слике губимо многе информације, а то није пожељно).

Сажимање

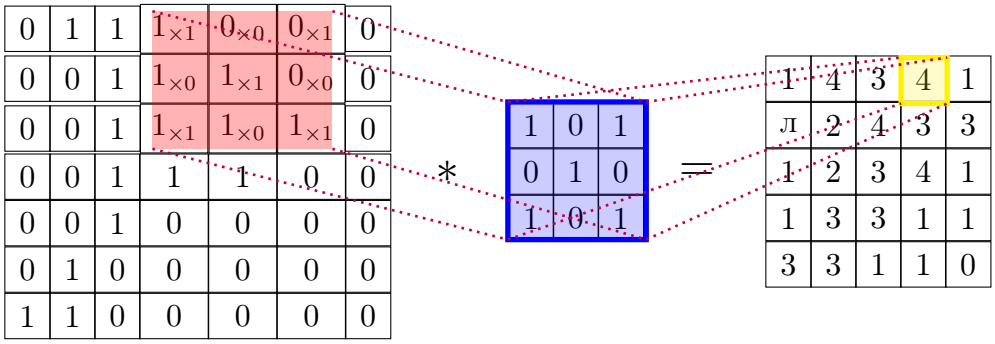
Сажимање као поступак користимо када желимо да смањимо величину слике. У зависности од типа сажимања бира се пиксел са одређеном карактеристиком у одређеном региону. Једно од познатих и најкоришћенијих сажимања јесте сажимање максимумом (енг. Max Pooling). У овом поступку памтимо максимуме одређених региона. Такође, познат је и поступак сажимања помоћу средње вредности (енг. Average Pooling). Овај поступак је аналоган сажимању максимумом. Разлика је у томе што се уместо максималне вредности рачуна средња вредност.



Пример сажимања

3.0.1 Конволуцијски слој

Конволуцијски слој је основна јединица грађе конволуцијске неуралне мреже. Овај слој обавља захтевне рачунице, али је и основа за



Поступак конволуције: Матрица 7×7 на коју се примењује филтер 3×3, где је корак $S = 1$

тренирање неурона неуралне мреже.

У општем случају нека је I квадратна матрица димензија $n \times n$ и кернел димензија $m \times m$ где је S корак. Нека је матрица добијена од матрице I применом кернела димензија $a \times a$, тада важи следећа релација:

$$a = \left[\frac{n-m}{S}\right] + 1$$

Операцију коволуције желимо приказати алгебарским изразом:

$$Y_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{ij}^{(l)} Y_j^{(l-1)}$$

где:

$Y_k^{(l)}$: излаз за I_l конволуцијски слој

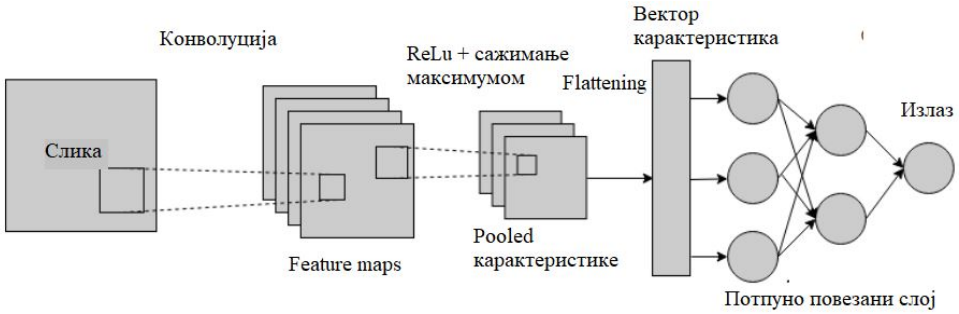
$b_i^{(l)}$: тренинг биас за I_l конволуцијски слој

$K_{ij}^{(l)}$: матрица која је филтер димензија $2h_1^{(l)+1} \times 2h_2^{(l)+1}$

Резултат једне конволуције се назива feature map. Међутим, врло често се дешава да имамо више филтера које треба применити на наш улаз. Тада од $2D$ улаза добијамо $3D$. Када се обаве и остале операције над тим $3D$ скупом feature map-а врши се операција *flattening*. Она $3D$ враћа у $2D$ у облику вектора карактеристика.

3.0.2 Потпуно повезани слој

У овом кораку вектор карактеристика представља улаз у неуралну мрежу. Где је потпуно повезани слој сличан скривеном слоју, једина разлика је што су у овом случају сви неурони једног слоја повезани са свим неуронима наредног слоја. Излаз овог слоја је заправо крајњи излаз, односно тражено решење.



Пример конволуцијске мреже

4 Учење неуралне мреже

Како би наша неуронска мрежа могла решевати неки конкретан проблем потребно је подесити њене тежине и биас како би за узорке њен излаз био тачан и решење оптимално. Када се неуралн мрежа тренира све његове тежине и биаси се на почетку случајно генеришу. Тренинг мреже почење од улазног слоја, иде кроз средишње слојеве све до излаза. Током тренинга се тежине и биаси мењају све док се не достигне да се за тренинг улазе као излаз не добија тачно решење. Један од најпознатијих метода који се користи јесте Backpropagation.

4.1 Алгоритам Bakpropagation

Када посматрамо неуронске мреже које као активациону функцију имају линеарну функцију оне могу описати само линеарне односе. Међутим, врло често је потребно описати и високо нелинеарне функције и тада заправо треба да користимо активационе функције које нису линеарне.

Идеја овог алгоритма јесте проналажење минималне грешке бирања тежина које су дефинисане у неуронској мрежи. Такође, комбинација

тежина која даје минималну грешку представља оптимално решење.

Како бисмо могли да се служимо овим алгоритмом потребно је да се упознамо са појмовима функције грешке и метода опадајућег градијента (енг. gradient descent).

Дефинишимо следеће променљиве:

$\vec{x_i}$: улаз
$\vec{y_i}$: жељени излаз за улаз x_i
$\vec{y_{1i}}$: вредност излаза неуронске мреже за улаз x_i
θ : ажуриране вредности биаса и тежина
$X = \{(\vec{x_1},\vec{y_1}), (\vec{x_2},\vec{y_2}) \dots (\vec{x_n},\vec{y_n})\}$
N : укупан број улаза
w_{ij}^k : тежина за неурон j у слоју I_k полазећи од неурона i
b_i^k : биас неурона i у слоју I_k
a_i^k : активација за неурон i у слоју I_k
o_i^k : излаз за неурон i у слоју I_k
r_k : број неурона у слоју I_k
σ : сигмоидна функција
g : активациона функција за скривене слојеве
g_o : активациона функција за излазни слој
η : стопа учења

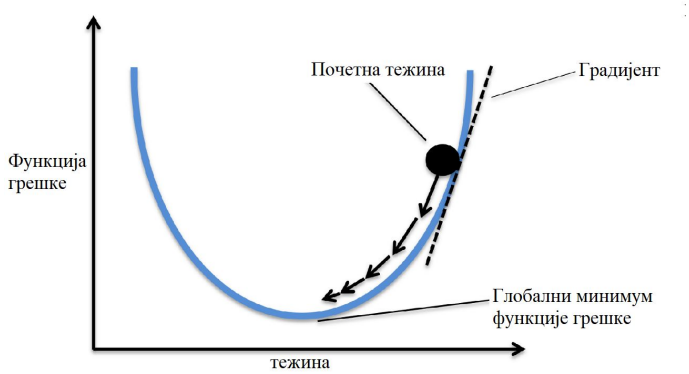
4.1.1 Функција грешке

Приликом варирања вредности за тежине и биас може доћи до одређене грешке. Желимо да да нађемо алгоритам који ће нам омогућити да пронађемо тежине и биасе тако да излаз мрежеза све уносе x тежи $y(x)$. Како бисмо остварили тај циљ, дефинишимо функцију губитка (енг. cost function). Једна од прихваћених функција губитка је следећа:

$$C(X, \theta) = \frac{1}{2N} \sum_x \|y(x) - y_1(x)\|^2 \tag{1}$$

Циљ алгоритма неуронске мреже и јесте да минимизује вредност функције грешке $C(X, \theta)$ који је у функцији од тежине и биаса. Како бисмо решили проблем минимизације користимо алгоритам познат као gradient descent.

4.1.2 Gradient descent



Визуелизација Gradient descent-а

При рачунању градијента прво се срачуна грешка последњег слоја. Након тога се дата грешка пропагира на претпоследњи слој где се одређује како је сваки неурон утицао на грешку. Потом се одређују компоненте градијента и ажурира вредност тежине у зависности од израчунатог градијента.

Нека је w_k тежина која спаја два неурона суседних слојева. Тада важи:

$$w_k{}' = w_k - \eta \frac{\partial C}{\partial w_k}$$

где:
$w_k{}'$ - ажурирана вредност тежине
η - параметар којим одређујемо брзину учења

Параметар учења је важно одабрати пре почетка тренирања, један од разлога је меморија.

4.1.3 Циљ алгоритма

Како бисмо могли да користимо дати алгоритам потребно је да буду задовољени наредни услови:

- Не постоје два неурона истог слоја која су повезана, неурон може бити повезан само са неуронима наредног слоја.
- Функција грешке зависи од процењеног и жељеног излаза. Такође, функција је непрекидна и диференцијабилна.

Backpropagation алгоритам одређује градијент сваког унутрашњег перцептрона у мрежи.

За функцију грешке користимо дати израз (1). Како бисмо олакшали математички рачун, нека важи:

$$b_i^k = w_{0i}^k$$

Тада важи:

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ji} o_j^{k-1} = \sum_{j=0}^{r_{k-1}} w_{ji} o_j^{k-1}$$

Такође, важи следећа једнакост:

$$\frac{\partial C(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial C_d}{\partial w_{ij}^k}$$

Одатле добијамо:

$$\boxed{C_d = \frac{1}{2}(y(x_d) - y_1(x_d))^2} \tag{2}$$

што представља грешку између очекиваног и израчунатог излаза.
Користећи се правилом извода сложене функције добијамо:

$$\frac{\partial C}{\partial w_{ij}^k} = \frac{\partial C}{\partial a_j^k} \frac{\partial a_j^k}{\partial w_{ij}^k} \tag{3}$$

Први чинилац горњег израза се често назива грешка и означава се са:

$$\delta_j^k = \frac{\partial C}{\partial a_j^k} \tag{4}$$

Нека је:

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial (\sum_{l=0}^{r_{k-1}} w_{il}^k o_l^{k-1})}{\partial w_{ij}^k} = o_i^{k-1} \tag{5}$$

Користећи (2), (3), (4) добијамо:

$$\boxed{\frac{\partial C}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1}} \tag{6}$$

Важно је нагласити да је дата парцијална деривација израчуната без разматрања функције грешке и активационе функције. Међутим, δ_j^k

тек треба да срачунамо, а она зависи од функције грешке C. Класичан backpropagation алгоритам користи као активациону функцију Сигмоидну функцију, а функција грешке је дата изразом (2).

Рачун грешке δ_j^k је показао да зависи од вредности грешака чланова следећег слоја. Из тог разлога рачун грешке рачунамо уназад од излазног слоја до улазног.

Излазни слој

Нека је I_m последњи слој. Функција грешке за тај слој гласи:

$$C = \frac{1}{2}(g_o(a_1^m) - y)^2 \tag{7}$$

Такође, важи:

$$\delta_1^m = (g_o(a_1^m) - y)g_o'(a_1^m) = (y_1 - y)g_o'(a_1^m) \tag{8}$$

Из (6), (7), (8) добијамо:

$$\boxed{\frac{\partial C}{\partial w_{ij}^m} = (y_1 - y)g_o'(a_1^m)o_i^{k-1}} \tag{9}$$

Скривени слојеви

Уочимо да важи следећа релација за $1 < m$:

$$\delta_j^k = \frac{\partial C}{\partial a_j^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial C}{\partial a_l^{k+1}} \frac{\partial a_j^{k+1}}{\partial a_j^k} \tag{10}$$

$$\implies \delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} \frac{\partial a_j^{k+1}}{\partial a_j^k} \tag{11}$$

Такође, по дефиницији важи и следећа једнакост:

$$a_l^k = \sum_{j=1}^{r^k} w_{jl}^{k+1} g(a_j^k) \tag{12}$$

Одатле закључујемо:

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1} g'(a_j^k) \tag{13}$$

Из (10), (11) добијамо израз познат као backpropagation формула:

$$\boxed{\delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1} g'(a_j^k) = g'(a_j^k) \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1}} \tag{14}$$

Добијамо:

$$\frac{\partial C}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = g'(a_j^k) o_i^{k-1} \sum_{l=1}^{r^{k+1}} w_{jl}^k \delta_l^{k+1} \tag{15}$$

Комбинујемо индивидуалне градијенте за сваки улаз-излаз пар $\frac{\partial C}{\partial w_{ij}^k}$ како бисмо добили укупни градијент $\frac{\partial C(\theta,X)}{\partial w_{ij}^k}$ за скуп улаз-излаз парова X:

$$\boxed{\frac{\partial C(\theta, X)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial C_d}{\partial w_{ij}^k}} \tag{16}$$

Знамо да једначина за ажурирање тежина гласи:

$$\boxed{{w'_{ij}}^k = -\eta \frac{\partial C(\theta, X)}{\partial w_{ij}^k}} \tag{17}$$

4.1.4 Кораци у алгоритму

1. За сваки улазно-излазни пар $(\vec{x}_i, \vec{y}_i)$ чувамо $y_1(x_i), a_j^k$ и o_j^k за сваки неурон j у слоју I_k почећи од слоја 0, улазни слој, до слоја m, излазни слој.

2. За сваки улаз-излаз пар $(\vec{x}_d, y_d)$ рачунамо и чувамо вредност $\frac{\partial C}{\partial w_{ij}^k}$ за свако w_{ij}^k које повезује неурон i у I_{k-1} слоју са неуроном j у слоју I_k све до последњег слоја I_m

3. (а) Рачунање грешке за последњи слој σ_1^m користећи једначину (8).

   (б) Рачунање грешке за скривене слојеве σ_j^k, крећући се позади од последњег скривеног слоја за који је $k = m - 1$, поновљеним коришћењем једначине (14).

   (ц) Израчунамо парцијални извод за сваку грешку C_d с обзиром на w_{ij}^k користећи једначину (16).

4. Комбинујемо индивидуалне градијенте за сваки улаз-излаз пар $\frac{\partial C}{\partial w_{ij}^k}$ како бисмо добили укупни градијент $\frac{\partial C(\theta,X)}{\partial w_{ij}^k}$ за скуп улаз-излаз парова X

5. Ажурирамо тежине према стопи учења η користећи једначину (17).

4.1.5 Одабир стопе учења η

Стопа учења, као што је већ речено, означава брзину и прецизност којом наша мрежа учи. Видели смо да је промена тежине пропорционална стопи учења, самим тим она има важну улогу у процесу учења, али и током процеса backpropagation. Из тог разлога је важно одабрати одговарајућу вредност за овај параметар. Показано је да је најбоље одабрати вредност из опсега:

$$0.01 \leq \eta \leq 0.9$$

Тачна вредност коју треба одабрати варира од проблема до проблема, тренинг скупа података итд.

5 Изазови који се јављају приликом тренирања неуралне мреже

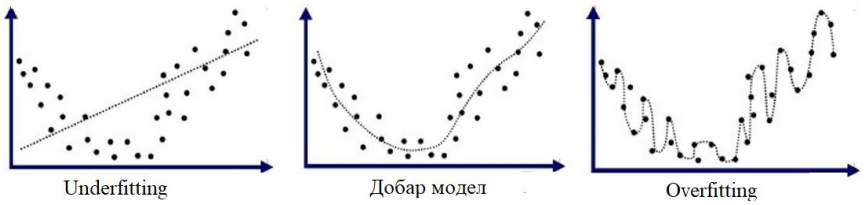
Идеалан модел представља најбољи могући перформанс неуралне мреже. Како бисмо проверили валидност модела ми користимо тренинг и тест сетове. Међутим, поред њих постоји и сет валидације. Он служи за подешавање хиперпараметара (објашњено у даљем тексту). Ова метода је нешто између тренирања мреже и тестирања.

Најједноставнији приступ претраге најбољег од више модела мрежа, чији је задатак исти, јесте рачунање функције губитка за податке који не спадају у тренинг скуп. На тај начин поредимо добијене резултате од више различитих мрежа које су прошле процес тренинга и она која има најмању функцију грешке у односу на валидациони скуп података представља наш најбољи модел.

5.1 Underfitting

Underfitting је појава да је грешка тренинг сета података значајно већа од очекиване грешке идеалног модела. Одакле закључујемо да је прецизност модела ниска и функција губитка велика.

То се приписује недовољној сложености нашег модела, односно олакшаном моделу него што јесте. Ми повећавамо сложеност модела повећањем броја слојева, неурона итд. Такође, некада је потребно дати више времена моделу за тренинг како би достигао свој најбољи перформанс, самим тим неодостатак времена исто може бити један од разлога underfitting-a.



Графички приказ Overfitting-a,Underfitting-a и доброг модела

5.2 Overfitting

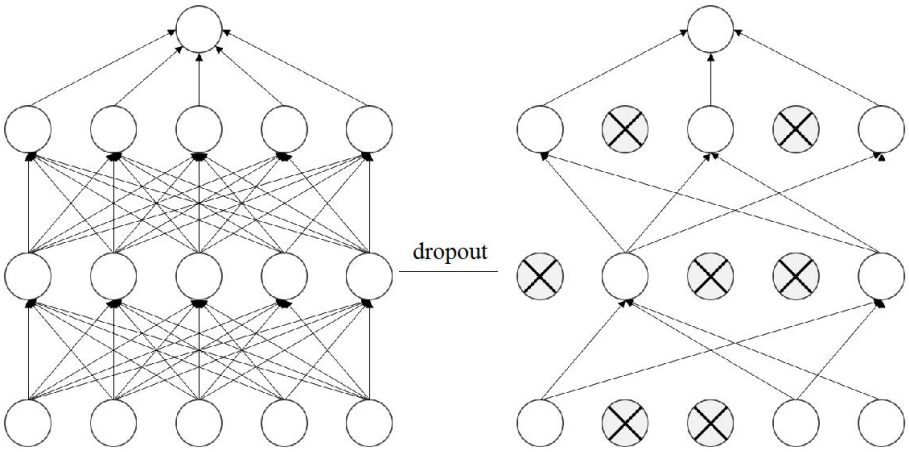
Overfitting је ситуација када наш модел уместо да учи статистичке правилности које су карактеристичне за наше тренинг податке заврши са меморисањем ирелевантних шумова уместо правог сигнала. Самим тим мрежа има го много лошији перформанс на новом скупу података у поређењу са тренинг скупом. Један од главних изазова машинског

Пример Augmentation методе

учења јесте минимизовање те грешке. Једно од решења, уједно и најбоље, јесте да користимо већи тренинг сет података. Зависно од проблем сета се може проширити на различите начине, на пример у случају да су улаз слике додавањем ротираних и рефлектованих слика улаза можемо проширити сет. Такође, могуће је додавати шумове, мењати контраст, боју, осветљење... То се назива Augmentation метода.

5.3 Dropout метода

Dropout је такође једно од решења. Врло често се дешава да одређени перцептрони мреже обављају више посла од осталих. Проблем се јавља када ти активнији перцептрони не науче добро функцију коју апроксимирају и греши, тада ће грешити и цела мрежа. Параметар ове је вероватноћа p. Током тренинга у различитим постају неактивни различити перцептрони са вероватноћом 1-p. Након сваког тренинга се сви ти неурони враћају у мрежу са непромењеним коефицијентима. За улазне и излазне неуроне је вероватноћа p веома мала и обично се



Приказ Droput методе

не избацују, док је за остале неуроне обично једнака 0,5. На тај начин неурална мрежа се адаптира да реши грешке прошлих слојева. Такође, важно је напоменути да је dropout активан само током тренинга. Овај поступак најчешће врши након активационог слоја.

5.4 Нестајући градијент и експлодирајући градијент

Приликом пропагације уназад градијента ажурирања његова вредност се може умањивати или увећавати уколико постоји функција на слојевима.

У случају да се тај градијент драстично смањује док се шири уназад кроз мрежу, може доћи до ситуације да грешка буде толико мала, готово занемарљива, близу улаза. Та појава се назива нестајући градијент која успорава и отежава учење и тренинг мреже.

Експлодирајући градијент се односи на ситуацију да се вредност градијента драстично повећава ширењем уназад. Тада долази до немогућности коришћења информација тог градијента.

6 Начини побољшања учења неуралне мреже

6.1 Иницијализација

Пре почетка обучавања мреже вредности параметара мреже се најчешће бирају насумично са већ одабраном дистрибуцијом. Овим је онемогућено да се више различитих неурона обучи за исти задатак, јер је циљ да сваки неурон научи различите функције њихових улаза. Овај поступак спречава стварање симетрије и омогућава правилно ажурирање података током алгоритма Backpropagation. Неке од познатих иницијализација су:

- Гаусова насумична иницијализација
- Униформна насумична иницијализација
- Ортогонална насумична иницијализација

6.2 Стохастични опадајући градијент

Стохастични опадајући градијент је оптимизациони алгоритам који служи за минимизацију функције губитка у односу на тренинг податке. Оно ажурира тежине за сваки елемент тренинг скупа појединачно

за разлику од backpropagation алгоритма који то врши као суму ажурирања целог тренинг скупа. Ова метода се показала као рачунски ефикаснија. Такође, показано се у овом моделу јавља стабилнија конвергеницица ка минимуму. Међутим, ова метода се најчешће користи у комбинацији са backpropagation алгоритмом. Предност је што ова техника користи мање ресурсе по ажурирању, самим тим погодује тренирању неуралних мрежа са великим тренинг скупом података. Овај алгоритам је базиран на традиционалним претпоставкама у оптимизационим проблемима. Међутим, те претпоставке неће увек важити код неуралних мрежа и то је мана ове технике.

6.3 Хиперпараметри

Хиперпараметрима се сматрају оним параметрима чију вредност користимо за конролисање процеса учења неуралних мрежа. То су број неурона у слоју, брзина учења, одабир активационе функције, број слојева итд. Вредности ових параметара су већ одређене односно познате пре почетка процеса учења. Различити алгоритми тренирања мреже захтевају различите хиперпараметре. Како они умногоме одређују рад наше неуралне мреже важно је подесети их на одговарајуће вредности. Такође, у већини случајева хиперпараметри су назависни једни од других. Самим тим је у тим случајевима могуће бирати оптимизоване хиперпараметре појединачно независно од осталих. Углавном се одабир хиперпараметара врши низом експеримената ручних варијација вредности на основу перформанси неуралне мреже.

6.4 Нормализација узорка

Код дубоких неуралних мрежа мала промена у параметрима почетних слојева може довести до већих промена у дубљим слојевима, а самим тим и на перформансу саме мреже. Ова метода поставља узорак тако да му је очекивање једнако нула, а стандардна девијација 1. Међутим, у случају да су првобитно очекивање и стандардна девијација боља варијанта, тада се неуралнa мрежа враћа на ту вредност. На тај начин је омогућено брже учење оптималних параметара за сваки улазни неурон.

6.4.1 Имплементација

Пре свега се нормализују улази. Самим тим је осигурано да наша мрежа ефикасно научи параметре првог слоја. Када посматрамо други

слој он прихвата активације од првог слоја. На тај начин ће и други слој ефикасније учити параметре. Тако ће и остали слојеви учити ефикасније и сваки слој ће бити нормализован.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2$$

$$x_i^{norm} = \frac{x_i - \mu}{\sqrt{\sigma^2 - \epsilon}}$$

$$y = \gamma x_i^{norm} + \beta$$

ϵ - мали број који се додаје како не би дошло до дељења нулом
β, γ - параметри који се тренирају

6.5 Регуларизација

Регуларизација је техника која смањује overfitting или смањује одступање у нашој мрежи смањењем сложености. Она представља широк појам која уклања и умањује појединачна одступања на нивоу тренинг скупа. Идеја је да се користе мање тежине како бисмо лакше контролисали учење. Најчешће врсте ове технике су L_1 и L_2 регуларизација. На функцију грешке се додаје регуларизациони члан тако да се смањује одступање промена тежина при ажурирању и тако вредност промена тежина остаје приближно једнак нули.

Члан регуларизације L_1: $\frac{\lambda}{N}\sum_{i=0}^{N-1}|w|$

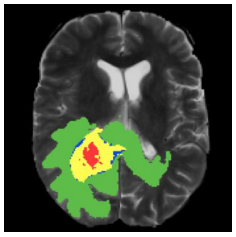
Члан регуларизације L_2: $\frac{\lambda}{2N}\sum_{i=0}^{N-1}w^2$

Где је λ коефицијент регуларизације којим се ограничава утицај регуларизационог члана на градијенте. Овај коефицијент се бира тако да мрежа учи главна обележја одређеног улаза, а не ирелевантна. Такође, овај коефицијент убрајамо у хиперпараметре.

7 Примена неуралних мрежа

Једна од најважнијих примена неуралних мрежа јесте у области обраде слике.

С развојем неуралних мрежа медицинска опрема за дијагностиковање која ради по принципима обраде слике је постала популарна и олакшала дијагностиковање односно напредак у медицинском третману. Слике у медицини које се широко користе су најчешће добијене путем магнетне резонанца (MRI), томографије (CT), X-зрака, ултразвука, позитронска емисијска томографија (PET)... Ове слике неретко служе лекарима за дијагностиковање различитих болести, самим тим је обрада слике постала један од фокуса рада у области компјутерских наука. Неки од примера су дијагностиковање тумора на мозгу, тумора коже, прелома костију, детектовање аутизма помоћу слике лица итд. Такође,



Приказ детектовања тумора на мозгу коришћењем обраде слике

обрада слике се користи у препознавању лица што има примену у различитим сферама попут идентификовања злочинаца. У агрикултури може служити за процену да ли је биљка здрава.

Још један пример примене неуралних мрежа јесу самовозећи аутомобили. У овом случају оне омогућавају детектовање пешака, саобраћајних знакова, семафорских светала, препознају путеве итд.

Самовозећи аутомобили

Примена у финансијама:

- Предвиђања на берзи

- Дозвола узимања кредита
- Откривање превара
- Прогноза цена
- Процена власништа

Индустрија:

- Контрола квалитета
- Оптимизација процеса производње
- Управљање процесима

Научна истраживања:

- Препознавање гена, протеина...
- Препознавање шаблона
- Моделирање физичких система

Ово су само неки од примера примене неуралних мрежа која је веома распрострањена. Из дана у дан све више напредује и очекујемо много нових изума у блиској будућности.

8 Закључак

У овом раду су дате основе принципа фукционисања напредујућих неуралних мрежа. Објашњено је на који начин ове мреже уче, честе грешке које се јављају при том процесу, али и методе оптимизације и побољшања њихове перформансе. Такође, упознали смо се и са широком применом неуралних мрежа.

Овом приликом бих желела да се захвалим свом ментору професору Петру Радовановићу као и свим осталим професорима који су ми предавали у Математичкој гимназији, уз које сам научила много и развила љубав према науци!

У Београду, мај 2021. Николија Бојковић

Литература

[1] Игор Шево: *Специјализована неуронска мрежа за класификацију и сегментацију аеро-снимака*, Бања Лука (2020).

[2] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, Debashis De): Fundamental Concepts of Convolutional Neural Networks

[3] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov: Dropout: a simple way to prevent neural networks from overfitting , Journal of Machine Learning Research, (2014).

[4] Backpropagation
https://brilliant.org/wiki/backpropagation/

[5] Introduction to Convolutional Neural Networks
https://www.kdnuggets.com/2020/06/introduction-convolutional-neural-networks.html/

[6] Samer Hijazi, Rishi Kumar, Chris Rowen): Using Convolutional Neural Networks for Image Recognition