**С.О.У. Гимназија „Мирче Ацев“ - Прилеп**



**Матурска проектна задача по Информатика**

**Тема: Вештачки невронски мрежи**

Изработил: Стефан Димески IV-7

Ментор: Анкица Стеваноска Јанческа

Датум: 15.02.2018

**Содржина**

**Вовед3**

**Вештачки невронски мрежи 5**

Градба на неврон 7

Поделба на Вештачките невронски мрежи според градбата 8

Feedforward 8

Конволуциони ВНМ 9

Автокодер11

**Gradient Descent – алгоритам за учење при надгледувано искуство**11

**Недостатоци кај алгоритмот Gradient Descent** 15

**Примена на ВНМ** 16

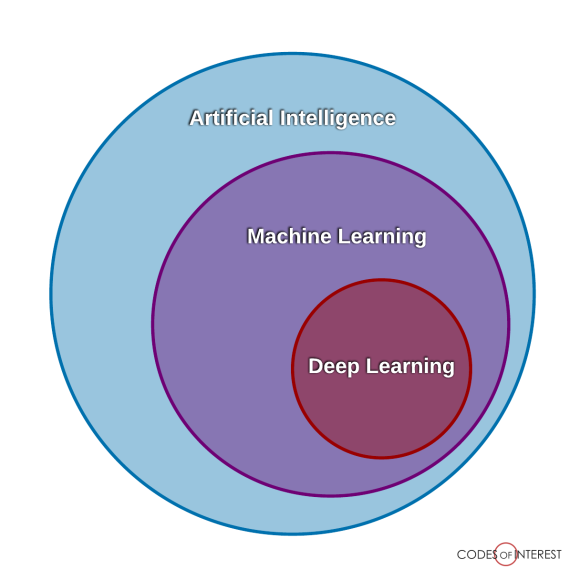
**Заклучок** 18

**Користена литература** 19

**Вовед**

Уште од појавата на првите компјутери научниците се обидувале да создадат софтвер кој ги има когнитивните капацитети на човечкиот мозок (вештачка интелигенција). Како научна дисциплина вештачката интелигенција се појавува во 1950-тите години и нејзина цел е истражување на интелигентни агенти ( т.е. секоја направа која може да ја перцепира својата околина и да превзема активности кои ја максимизираат нејзината шанса за успех во остварување на некоја цел. Но често терминот вештачка интелигенција се употребува за машини што ги имитираат когнитивните функции што луѓето ги поврзуваат со човечкиот мозок, како на пр. учењето и решавањето на проблеми. Набргу се сфатило дека проблемите кои му се тешки на човекот како на пример некои комплицирани ариметички калкулации, компјутерите лесно ги решаваат. Додека пак проблемите како на пример препознавањето на објекти на слика итн. кои се лесни за човекот (човекот ги решава подсвесно, без свесен напор), претставуваат голем проблем за компјутерите. Со други зборови, проблемите кои можат да се опишат преку листа на формални математички правила се тешки за човекот, но лесни за компјутерот. Проблемите кои не можат да се дефинираат на тој начин (формално, преку математички правила), на пример препознавањето на објекти, кои човекот ги решава „автоматски“, подсвесно, се тешки за компјутерот.

Првите обиди за создавање на вештачка интелигенција биле во форма на експертски системи (анг. Expert systems) кои се појавиле во 1970-тите години и имале цел да го симулираат процесот на носење одлуки на човек кој е експерт за проблемот што го решаваат (оттука доаѓа и името). Овие експертски системи имаат база на знаење која е создадена преку мануелно внесување на правила, а потоа преку неа компјутерт може да донесува логички заклучоци. Но, овој обид се покажал неуспешен и покрај добрите страни. Како главна причина се наведува тешкотијата да се внесуваат огромен број на правила мануелно и тешкотијата да се одржуваат експертските системи кои имаат големи бази на знаење.

Како втор обид се појавува машинското учење (анг. Machine learning) чија цел е создавање на програма која сама учи како да реши одреден проблем, за разлика од претходниот обид, каде сите правила се внесени од човекот/експертот. Машинското учење доаѓа со свои нови тешкотии, но ги решава главните проблеми на експертските системи. Во оваа дисциплина се јавуваат алгоритми како линеарна и логистичка регресија, дрво на одлуки(Decision Tree), Support Vector Machines итн. Се појавува и Перцептронот на Росенблат, кој подоцна ќе стане важен дел на други идеи. Потоа доаѓа Конекционизмот, движење кое во научната дисциплина Вештачка интелигенција се појавува во 1980-тите. Главната идеја на Конекционизмот е дека голем број на едноставни процесни единици можат да постигнат интелигентно однесување кога се поврзани меѓусебно. Пример е човечкиот мозок, кој се состои од огромен број на мали процесни единици-неврони, кои се поврзани меѓусебно. Ова движење одиграло голема улога при формулирањето на **вештачките невронски мрежи** кои спаѓаат под дисциплината машинско учење**.**

**Вештачки невронски мрежи**

За разлика од првичните, денешните вештачки невронски мрежи се прилично напреднати, така што доаѓаат во многу форми, но основната форма претставува неколку слоеви од неврони кои се меѓусебно поврзани. Идејата за ваквата архитектура доаѓа од начинот на кој е изграден мозокот. Постојат повеќе видови на невронски мрежи како на пр: Feedforward, Feedback, Конволуциони (Convolutional NN), Периодично-Повратни (Recurrent NN), Долготрајна-краткотрајна меморија (Long-Short term Memory – LSTM), Автокодер, Длабоко-верување(Deep Belief NN)..... кои се разликуваат според нивната структура (архитектура), намена, начинот на „учење“, правецот на течење на информациите низ слоевите и индивидуалните неврони.

Во машинското учење, алгоритам кој учи е составен од 3 дела: Задача, мерка на успех и искуство.

**Задачата** претставува проблемот кој сакаме да го решиме со алгоритмот. Многу е важно да се напомене дека „учењето“ не е задачата на ваквите алгоритми, туку тоа е само начинот на кој алгоритмот ги добива вештините за решавање на задачата. Ако сакаме да му додадеме можност на некој робот да се движи, тогаш можеме или директно и мануелно да го програмираме роботот да оди, или пак да да го програмираме роботот да може да учи да оди и во тој случај задачата е вештината на одење. Задачите се делат на повеќе групи, од кои најважни се:

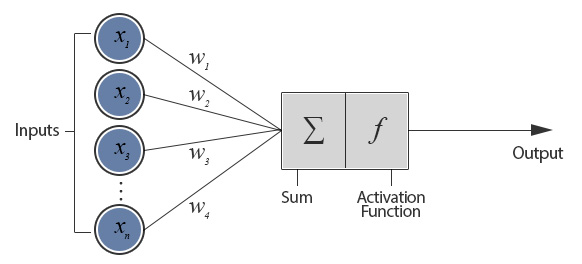
* **Класификација**. Во овој тип на задачи, програмата добива некој влез кој треба да го класифицира во некој број на категории. Со други зборови, програмата за излез треба да прикаже во која категорија спаѓа влезот. Пример за класификација е препознавањето на објекти: програмата за влез ги добива пикселите на некоја слика, а како излез ја прикажува веројатноста објектот на сликата да припаѓа на некој од дадените категории.
* **Регресија**. Во овој тип на задачи, програмата треба да предвиде некоја нумеричка вредност, која зависи од величината која се дава во влезот на програмата. Пример за регресија претставува програма која предвидува за колку пари ќе се продаде некоја куќа, која за влез ја добива нејзината големина.
* **Транскрипција**. Пример за овој тип претставува програма која како влез ги добива пикселите на некоја слика на која има текст, а програмата го дава текстот во форма на низа на карактери.
* **Преведување**. Во овој тип на задачи, програмата за влез добива текст на еден јазик, а за излез треба да го преведе текстот на друг јазик.
* **Откривање** **на** **аномалии**. Во овој тип на задачи, програмата разгледува голем број на податоци и ја пријавува секоја аномалија која ја забележува. Пример за овој тип на задача е програма која детектира можност за украдена кредитна картичка, преку тоа што забележува промена (аномалија) во начинот на трошење на парите или промена во тоа што е купувано.
* **Синтеза**. Во овој тип на задачи, програмата треба да генерира нови примери, кои се слични на примерите од примерите за тренинг.

Со цел да ги одредиме вештините на еден алгоритам на машинското учење, мораме да имаме некаква квантитативна мерка која ќе го мери неговиот перформанс т.е. колку добро ја извршува задачата. За ова е задолжен вториот дел од алгоритмот – **мерката за успех**, која е најчесто различна за различни задачи. При извршување на некоја задача од страна на алгоритмот, може да се мерат различни вредности, како на пр. прецизноста која претставува пропорцијата на точно извршени задачи од сите задачи кои ги извршува алгоритмот. Мерката за успех е клучна за „учењето“ на алгоритмот, бидејќи ги дава потребните квантитативни информации за успехот во извршувањето на задачата, со цел алгоритмот да има претстава како да се подобри т.е. да „учи“. Одредувањето на мерка за успех не е секогаш едноставно, бидејќи често постојат најразлични мерки кои можат да се изберат, а кои даваат различни крајни резултати при „учењето“. Исто така постојат и величини кои идеално би го мереле перформансот, но не може да се измерат, па се користат мерки кои го приближно ја прикажуваат таа мерка.

Алгоритмите на машинското учење можат исто така да се делат и според **искуството** (третиот дел) кое го добиваат во процесот на „учење“. Главно можат да се поделат на две категории: надгледувано учење и ненадгледувано учење, иако постојат и други типови на учење како: учење со појачување( Reinforcement Learning ), непријателско учење (Adversarial Learning), Online Learning, Q-Learning. Во надгледуваното учење, алгоритмот има пристап до низа од примери и низа од точни одговори за секој пример, додека во ненадгледуваното учење алгоритмот има пристап само до низа од примери, без точните одговори. Третиот важен тип на учење е учење со појачување, каде што алгоритмот нема пристап до фиксирана низа на примери, туку добива примери преку интеракција со околината.

Значи, вештачките невронски мрежи претставуваат еден вид на алгоритам на машинското учење кои можат да имаат најразлични задачи, мерки за успех и искуства (начини на учење). Во најголем број од случаите, секој тип на невронски мрежи може да се тренира со различните начини на учење, но за некои типови на невронски мрежи, одредени начини за учење се поефикасни од другите.

**Градба на неврон**

Вештачките мрежи се изградени од неврони, кои најчесто се групирани во слоеви. Невроните се инспирирани од биолошките неврони и имаат слична градба со нив. Секој неврон од еден слој е поврзан со секој неврон од претходниот слој од каде прима информации, кои после одредена преработка (применување на одредени математички операции) ги пренесува кон сите неврони од следниот слој. Информациите најчесто се броеви, а секој неврон како информација добива еден број. Секоја конекција меѓу невроните има своја тежина, која е клучниот дел од учењето. Тежините се тие кои се менуваат при процесот на учење. Невронот работи на следниот начин: излезот од секој неврон од претходниот слој кој е поврзан со тековниот неврон е помношен со соодветната тежина на конекцијата меѓу тие неврони. Потоа тие вредности се собираат и поминуваат низ одредена функција наречена активациска функција (анг. Activation function) како на пример сигмоидна или ReLU функција која дава вредност ограничена најчесто помеѓу 0 и 1. Излезот на оваа функција е информацијата (бројот) кој се пренесува на сите неврони од следниот слој.

C:\Users\User\Desktop\ql_288f3f843d30c7dbeb80aac352dd3e68_l3.pngИзлезот на секој неврон се пресметува со следната формула:

Каде S е множество составено од сите неврони од претходниот слој (l-1)

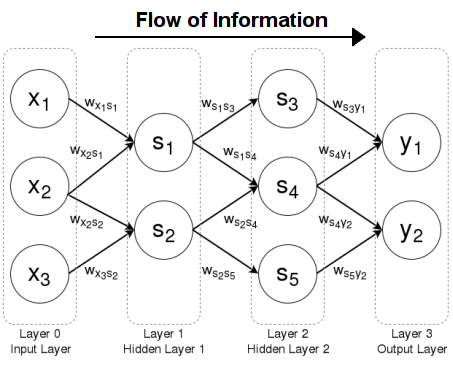
**Поделба на Вештачките невронски мрежи според градбата**

Како што видовме претходно, постојат најразлични градби/структури/архитектури на вештачките невронски мрежи. Најкорисни досега се: Feedforward, Feedback, Конволуциони (Convolutional NN), Периодично-Повратни (Recurrent NN), Долготрајна-краткотрајна меморија (Long-Short term Memory – LSTM), Автокодер, Длабоко-верување(Deep Belief NN).....

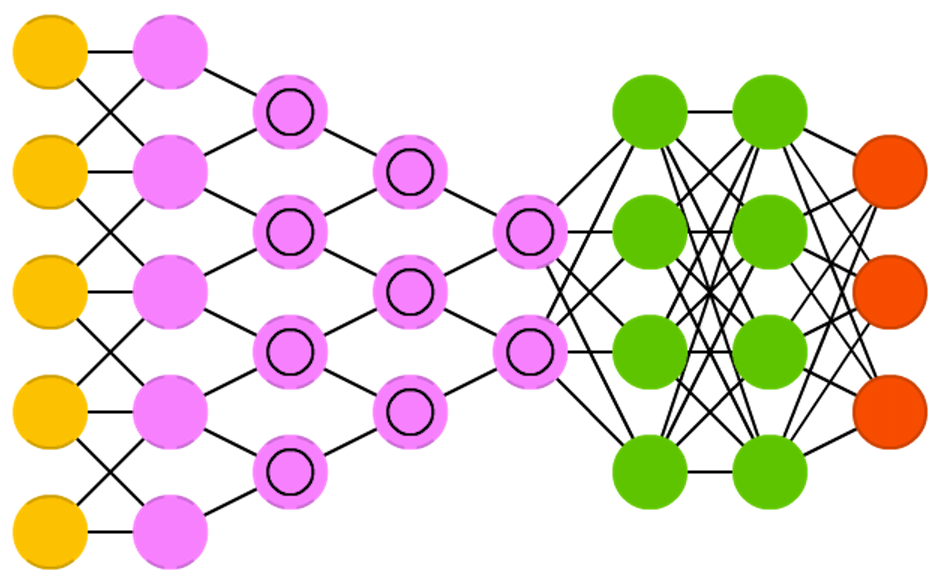
**Feedforward невронски мрежи**

Основниот тип на невронски мрежи се Feedforward невронските мрежи. Овие мрежи се состојат од слој од неврони кои ги добиваат податоците (влез), слој од неврони наречен скриен слој и трет слој кој го дава излезот на мрежата. Секој неврон од влезниот слој е поврзан со секој неврон од скриениот слој. Општо, секој неврон од еден слој, е поврзан со секој неврон од наредниот слој. Ваквите мрежи помеѓу слојот за влез и слојот за излез, можат да имаат и повеќе од еден скриен слој (Hidden layer), при што се нарекуваат длабоки невронски мрежи (анг. Deep Neural Networks). Додавањето на нови скриени слоеви со различен број на неврони ја зголемува способноста на невроската мрежа да учи покомплексни проблеми. Не е необично да се употребуваат големи невронски мрежи со преку 10000 неврони.

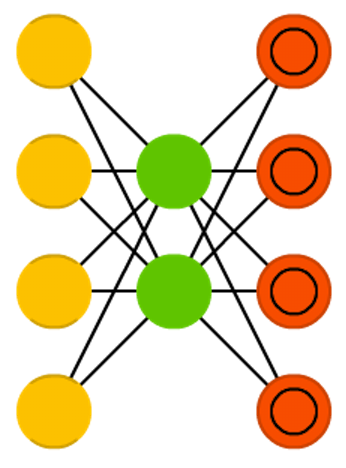
Feedback невронските мрежи се слични со Feedforward невронските мрежи, со таа разлика што во Feedforward невронските мрежи, конекциите меѓу невроните можат да бидат и со обратна насока т.е. некои информации да течат од одреден слој кон претходниот слој на мрежата. Се разбира, поради покомплексната структура на овој тип, процесот на тренирање („учење“) е потежок и подолготраен.



**Конволуциони невронски мрежи**

Конволуционите невронски мрежи содржат во нив Feedforward невронска мрежа (ФНМ), но пред оваа мрежа постојат одреден број на конволуциони слоеви каде информациите се преработуваат преку операцијата конволуција за потоа да се пренесат на Feedforward невронска мрежа која ги дава крајните резултати. Овој тип на невронски мрежи најчесто се употребува на слики и за разлика од ФНМ-те кај кои секој неврон од влезниот слој добива по еден пиксел од сликата ( за слика со димензии 256х256 би биле потребни 65536 неврони во влезниот слој ), во Конволуционите се внесуваат помали сегменти од сликата како на пример сегменти со големина 8х8 пиксели. Ова можеме да го замислиме како матрица со големина 8х8 која се движи низ сите можни позиции на сликата и за секоја позиција имаме нова итерација на невронската мрежа. Но покрај сето тоа, оваа 8х8 матрица претставува конволуциона операција, која секој пиксел го модифицира на одреден начин за да се постигне некоја цел. На пример постои матрица која кога ќе помине низ целата слика се добива нова слика на која се е со црна боја освен рабовите од десно или пак која може да додаде ефект како на пример заматување итн. Моќта на Конволуционите невронски мрежи е во тоа што повеќе вакви конволуциони операции се прават една по друга (на резултатот од едната, се применува другата), па идејата е: со првите конволуции се препознаваат рабови, со вторите – контурите на објектите и со секоја понатамошна конволуција се добиваат се по апстрактни репрезентации, кои на пример кулминираат со препознавањето на лице на сликата. Потоа овие крајни репрезентации се предаваат на ФНМ каде се врши препознавањето. Параметрите на секоја конволуциона операција можат да се менуваат, со што мрежата може да научи кои параметри се најдобри според тоа што се бара од неа.

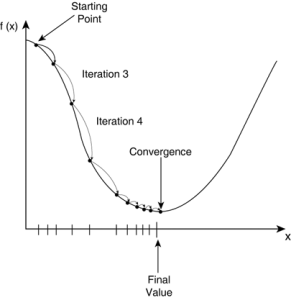
**Автокодер**

Автокодерите се невронски мрежи со посебна намена: учење на покомпактна, подобра и поинформативна репрезентација на податоците кои ги добиваат за влез. За таа цел овие мрежи започнуваат со одреден број на невронски мрежи во влезниот слој, а во натамошните слоеви овој број се намалува за некој фактор се до средината каде имаме најмалку неврони. Потоа бројот на невроните во следните слоеви почнува да се зголемува по истиот начин на кој претходно се намалуваа. Всушност овие невронски мрежи можат да се поделат на две: кодер (првата половина заедно со средината) и декодер (втората половина заедно со средината). Автокодерот успешно ја завршил задачата доколку вредностите на влезниот и излезниот слој за различни влезови се исти, а тоа значи дека во средината се состои најкомпактната и најинформативната репрезентација на податоците од влезот. Кодерот информациите ги претвора во компактната форма, а декодерот од компактната форма ги враќа во оригиналната форма. После завршувањето на тренингот, овие два дела се разделуваат и претставуваат две посебни невронски мрежи.

Автокодерите најчесто се употребуваат за добивање на компактна репрезентација на податоците ( компресија ) и како додатоци на други мрежи.

**Gradient Descent – алгоритам за учење при надгледувано искуство**

Секој вид на искуство на алгоритмите на машинското учење има различни алгоритми кои го извршуваат „учењето“ (тренингот). За надгледуваното искуство најупотребуван ваков алгоритам е Gradient Descent.

Всушност учењето во ваквиот вид на искуство претставува оптимизирање на некоја таканаречена објективната функција. Објектнивната функција е всушност мерката за успех и со нејзино оптимизирање ( барање на глобален минимум/максимум ) го извршуваме учењето. Оваа функција како параметри ги добива тежините на конекциите меѓу невроните, излезот на мрежата и посакуваниот излез на мрежата. „Учењето“ е всушност менувањето на вредностите на тежините на конекциите, со цел да се оптимизира објективната функција. Оптимизацијата на оваа објективна функција се врши преку алгоритмот **Gradient Descent**.

Овој алгоритам работи на следниот начин: во невроска мрежа се внесуваат влезните податоци и се проверува успехот преку објективната функција (мерката за успех). Доколку мрежата не била успешна т.е. има грешки, алгоритмот дознава како да ги менува тежините на конекциите со цел да ги минимизира грешките или со други зборови да го максимизира успехот ( во практиката, објективната функција најчесто ја мери големината на грешките, а не на успехот, па затоа таа најчесто се минимизира ). Како да ги менува тежините алгоритмот дознава преку земањето на вредноста на парцијалниот извод на објективната функција со почит кон секоја од тежините, па ги променува нивните вредности според него. Горенаведените чекори се повторуваат се додека мрежата повеќе не прави грешки. Барањето на вредноста на парцијалниот извод на ваква сложена објективна функција не е едноставно, па за тоа се користи друг алгоритам наречен назадна-пропагација на грешки (анг. Backpropagation of errors). Често луѓето прават грешка сметајќи дека назадната-пропагација на грешки е алгоритмот кој го врши учењето. Назадната-пропагација на грешки (која е посебна примена на погенералниот метод на компјутерска диференцијација наречен Автоматска диференцијација во обратна состојба – Reverse-Mode Automatic Differentiation) само ја наоѓа вредноста на парцијалниот извод, која се користи од алгоритмот Gradient Descent извршување на учењето.

Процесот на тренирање во форма на чекори:

Фаза 1 - Пропагација:

- се внесуваат вредностите на влезот на невронската мрежа и се пресметуваат вредностите на последователните слоеви вклучувајќи го и последниот слој (слојот за излез).

- пресметување на грешките со помош на објективната функција

- користење на назадната-пропагација на грешки за да се пропагираат грешките од последниот слој кон претходните слоеви т.е. да се определи колкави се грешките на претходните слоеви. Грешките се пропагираат наназад се додека не се стигне до влезниот слој.

Фаза 2 – Менување на тежините:

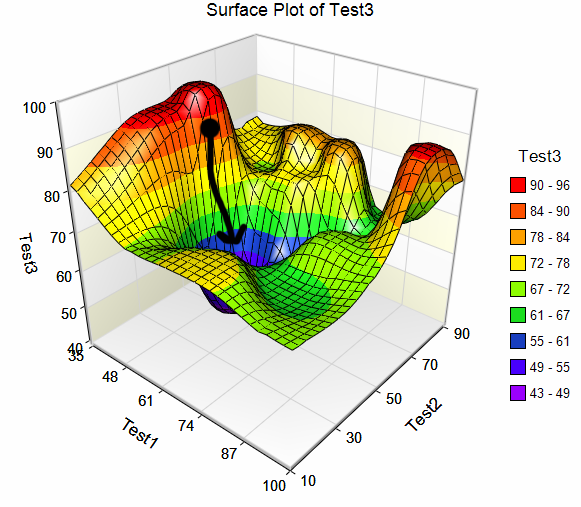
- се бараат вредностите на парцијалните изводи на објективната функција со почит кон секоја од тежините

- секоја од тежините се менува (зголемува/намалува) во насока која ја минимизира објективната функција. Оваа насока се добива од парцијалниот извод.

Овие две фази се повторуваат се додека грешките не се минимизираат или пак некој друг услов е исполнет.

****

**Недостатоци на алгоритмот Gradient Descent**

Најчесто во практиката, невронските мрежи кои се употребуваат содржат голем број на неврони организирани во голем број на слоеви. На пример доколку имаме една невронска мрежа составена од четири слоеви од кои првиот има 5 ****неврони, вториот има 8 неврони, третиот-3 неврони и последниот- 1 неврон, тогаш имаме вкупно 5\*8\*3\*1= 120 тежини кои можеме да ги изменуваме. Тоа значи дека објективната функција која како параметри ги има тежините и влезот на невронската мрежа има вкупно 125 параметри. Секој параметар претставува посебна димензија (бидејќи може да се менува), па можеме да кажеме дека објективната функција во овој пример е функција која мапира 125-димензионален простор во 1-димензионален простор т.е. :

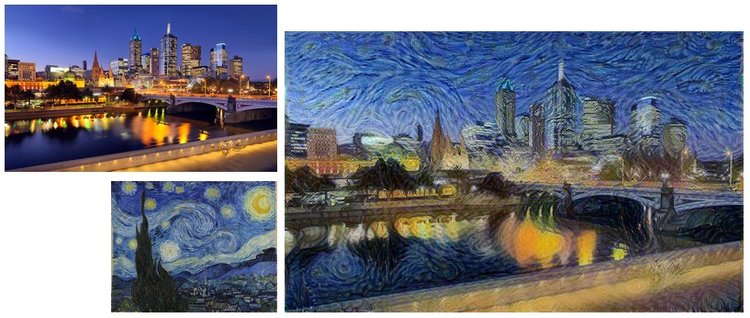
**C:\Users\User\Desktop\ql_49e930647745ce8af8f1633bdaf4de20_l3.png**

Изводите на оваа функција со почит кон секоја тежина ни кажуваат во која насока на 125-димензионалниот простор да се движиме со цел најефикасно да ја намалиме грешката. Но, најчесто ваквите функции не се едноставни за оптимизирање бидејќи нивната форма не е конвексна (парабола) каде има само еден глобален минимум и нема попатни локални минимуми. Во објективните функции најчесто има голем број на локални минимуми и неколку глобални минимуми. Идеално, ние би сакале нашиот алгоритам да пронајде некој глобален минимум за објективната функција да биде максимално оптимизирана. Но тука доаѓа до израз еден недостаток на алгоритмот Gradient Descent – овој алгоритам не може да разликува меѓу локалните и глобалните минимуми (бидејќи и во двата случаи парцијалните изводи се еднакви на нула) па алгоритмот заглавува во некој локален минимум, мислејќи дека го пронашол глобалниот минимум, при што се добиваат не-оптимални решенија на проблемот.

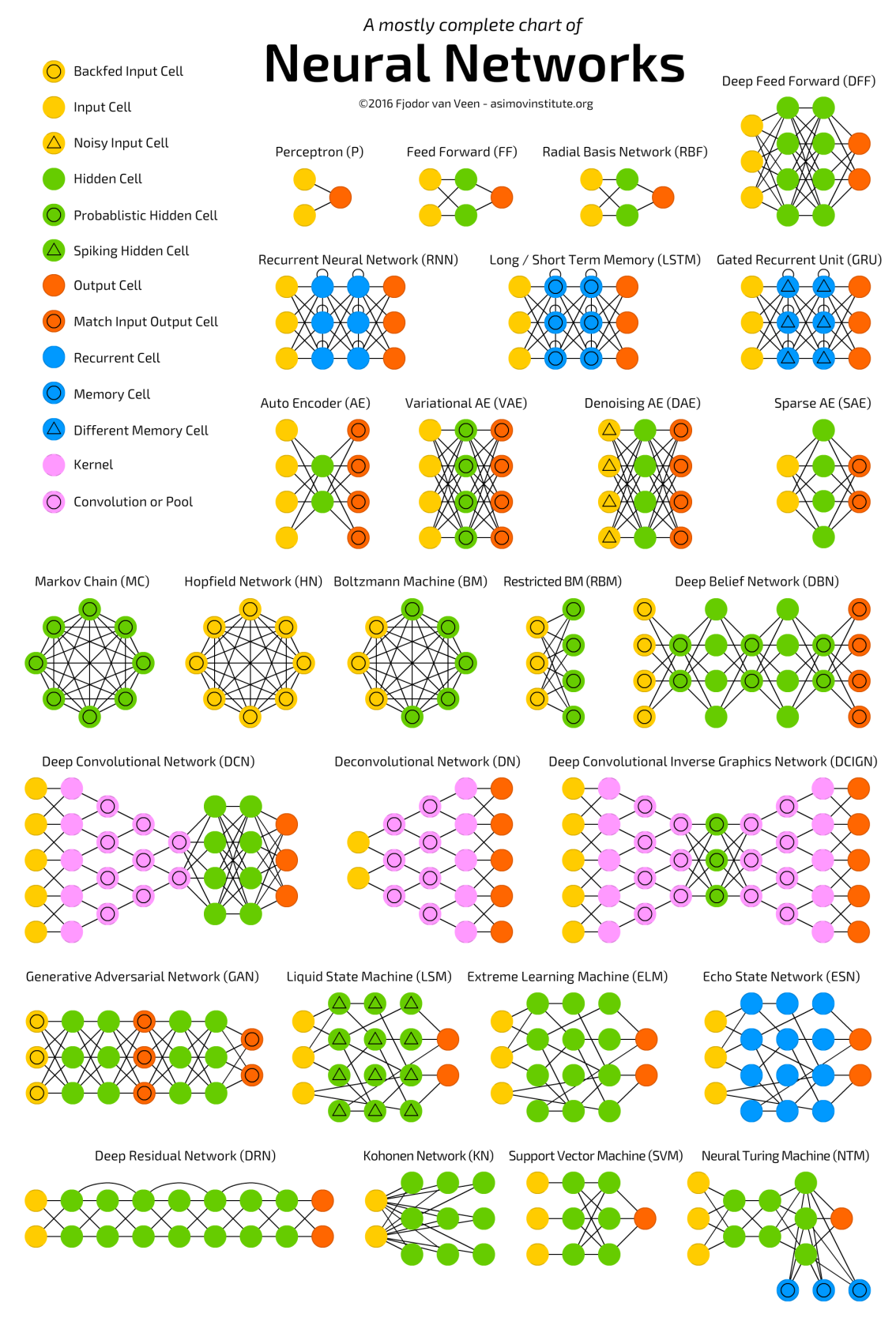
Постојат неколку начини на надминување на овој проблем, како на пример додавање на инерција, очекувајќи алгоритмот да може да излезе од локалниот минимум. Иако ваквите начини имаат периодичен успех, најчесто не се доволни и овој проблем сеуште останува нерешен.

**Примена на ВНМ**

Вештачките невронски мрежи денес имаат голема примена во сите сфери на информатиката. Конволуционите се употребуваат за препознавање лица или други објекти на слики во апликациите на мобилните телефони, за препознавање на рачно пишан текст или број. Периодично-повратните се користат за преведување од еден јазик на друг (Google Translate), за препознавање на говор (говорни команди) како на пример во апликациите Сири, Кортана или Гугл асистент.

Како визуелно атрактивна примена може да се спомне користењето на ВНМ за пренесување на стилот на една слика врз друга. На пример во мрежата се внесува фотографија, а за излез се добива фотографијата во стилот на некоја слика (на пример во стилот на слика на Ван Гог).

Вештачките невронски мрежи се користат и во автомобилите кои возат сами (без потреба за возач), каде ги препознаваат улиците, сообраќајните знаци и ленти и донесуваат одлуки како да се движи автомобилот. Се смета дека кога ваквите автомобили ќе се усовршат, тие ќе бидат најбезбедните автомобили, бидејќи како најчеста причина за сообраќајните несреќи се нагласуваат човечките грешки, кои со ваквите автомобили би се елиминирале.

****Можностите за примена на вештачките невронски мрежи се бескрајни.

**Заклучок**

Вештачките невронски мрежи направија вистинска револуција во компјутерските науки, овозможувајќи со огромна прецизност да се решат одредени проблеми кои претходно не можеле да се решат со други методи. Денес тие се едни од најпопуларните методи/алгоритми во сферата на вештачката интелигенција. Како сведок за тоа може да послужи фактот што секојдневно се објавуваат по 2-3 научни трудови поврзани со нив. Секоја година се одржуваат конференции за претставување на најновите иновации во полето на машинското учење, а рекордите за прецизност на натпреварите во препознавање на објекти се рушат веќе неколку последователни години од различни вештачки невронски мрежи. Иако е млада дисциплина во науката, машинското учење многу брзо напредува, а особено полето на вештачките невронски мрежи кое нуди неограничени можности за понатамошен развој.

Иако научниците не ги гледаат вештачките невронски мрежи како алгоритам кој самостојно ќе ја реши интелигенцијата (Artificial General Inteligence), сепак вреди да се истражува ова поле, кое досега вроди големи успеси и кое се очекува понатаму да продолжи во иста мера.

**Користена литература**

Deep learning - Ian Goodfellow, Yoshua Benigo, Aaron Courville; MIT Press, 2016

<https://en.wikipedia.org/wiki/Kernel_(image_processing)>

<https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks>

<https://en.wikipedia.org/wiki/Gradient_descent>

<https://en.wikipedia.org/wiki/Backpropagation>

<https://en.wikipedia.org/wiki/Machine_learning>

<https://en.wikipedia.org/wiki/Supervised_learning>

<http://www.asimovinstitute.org/neural-network-zoo/>

<https://en.wikipedia.org/wiki/Artificial_neural_network>

<http://neuralnetworksanddeeplearning.com/index.html>

<https://www.youtube.com/watch?v=bxe2T-V8XRs>

<https://www.youtube.com/watch?v=py5byOOHZM8>

<https://www.youtube.com/watch?v=aircAruvnKk>

<http://colah.github.io/posts/2015-08-Backprop/>

<https://en.wikipedia.org/wiki/Automatic_differentiation>