**Вештачки невронски мрежи**

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**.**

Вештачките невронски мрежи до денес 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 алгоритмот добива примери преку интеракција со околината.