Strojové učení

26. listopadu 2024

Strojové učení je podoblastí umělé inteligence, zabývající se algoritmy a technikami, které umožňují počítačovému systému 'učit se'. Učením v daném kontextu rozumíme takovou změnu vnitřního stavu systému, která zefektivní schopnost přizpůsobení se změnám okolního prostředí.

Strojové učení se značně prolíná s oblastmi statistiky a dobývání znalostí a má široké uplatnění. Jeho techniky se využívají např. v biomedicínské informatice (tzv. systémy pro podporu rozhodování), rozlišení nelegálního užití kreditních karet, rozpoznávání řeči a psaného textu či mnohé další. Algoritmy však mohou být tendenční.

Základní rozdělení algoritmů učení

Algoritmy strojového učení lze podle způsobu učení rozdělit do následujících kategorií:

- učení s učitelem (en: supervised learning) – pro vstupní data je určen správný výstup (třída pro klasifikaci nebo hodnota pro regresi)
- učení bez učitele (en: unsupervised learning) – ke vstupním datům není známý výstup
- kombinace učení s učitelem a bez učitele (en: semi-supervised learning) – část vstupních dat je se známým výstupem, ale další data, typicky větší, jsou bez něj. Často se používá EM algoritmus (en: Expectation–Maximization algorithm). Podobný přístup je transdukce

- zpětnovazebné učení (en: reinforcement learning), též učení posilováním

Podle způsobu zpracování lze algoritmy rozdělit na:

- dávkové – všechna data požadují před začátkem výpočtu
- inkrementální – dokážou se "přiučit", tj. upravit model, pokud dostanou nová data, a to bez přepočítání celého modelu od začátku

Základní druhy úloh strojového učení

- Klasifikace – rozděluje vstupní data do dvou nebo několika tříd
- Regrese – odhaduje číselnou hodnotu výstupu podle vstupu
- Shlukování – zařazuje objekty do skupin s podobnými vlastnostmi, typicky při učení bez učitele

Dalšími typy úloh jsou:

- Ranking určuje pořadí datových bodů, výsledkem je částečné nebo úplné seřazení
- Učení strukturovaných dat – výstupní neboli hledaná struktura může být například sekvence, strom, graf, matice ...

  Aplikacemi jsou např. učení syntaktických stromů ve zpracování přirozeného jazyka, zarovnání sekvencí proteinů v bioinformatice, převod řeči na textový řetězec, tj. na sekvenci znaků, hledání vhodné molekuly reprezentované jako graf v chemoinformatice, výstup obrázkového algoritmu jako matice apod. (mnoho konkrétních úloh) ...

Podoblasti strojového učení

- Používané modely:
  - Rozhodovací stromy
  - Algoritmus k-nejbližších sousedů
  - Podpůrné vektory, viz Support vector machines
  - Lineární diskriminační analýza (en: Linear discriminant analysis)
  - Kvadratická diskriminační analýza (en: Quadratic discriminant analysis)
  - Množina rozhodovacích pravidel
  - Perceptron
  - Bayesovské sítě
  - Neuronové sítě
- Techniky pro kombinaci více modelů (en: Ensemble learning a en: Meta learning (computer science))
  - Bootstrap aggregating (resp. zkratka Bagging)
  - Boosting
  - Stacking

- Testování přesnosti modelu:
  - Křížová validace
  - Bootstrap (statistika) (en:Bootstrapping (statistics)) a také
  - Odhady pravděpodobností správné klasifikace

Terminologie

- Data, body, případy, měření
- Atributy, rysy, proměnné, příznaky (features)
- Druhy/typy atributů: binární, kategoriální (např. "A", "B", "AB" nebo "0" pro krevní skupiny), ordinální (např. "velký", "střední" nebo "malý"), celočíselné (např. počet výskytů slova v emailu) anebo reálné (např. měření krevního tlaku); strukturované, hierarchické

Software

RapidMiner, KNIME, Weka, ODM, Shogun toolbox, Orange, Apache Mahout a scikit-learn jsou softwarové balíky, které obsahují různé algoritmy strojového učení
Online: Microsoft Azure Machine Learning (Azure ML)

Klasifikace metod strojového učení podle vynaloženého úsilí na získání nových znalostí

- **Učení zapamatováním** (*rote learning*, biflování)

  Pouhé zaznamenání dat nebo znalostí.
- **Učení se z instrukcí** (*learning from instruction*)

  Provádí se integrace se znalostmi již získanými.
- **Učení se z analogie** (*learning by analogy*)

  Zapamatování již vyřešených případů.
- **Učení na základě vysvětlení** (*explanation based learning*)

  Založeno na několika příkladech a rozsáhlých znalostech z dané oblasti.
- **Učení se z příkladů** (*learning from examples*)

  Založeno na velkém množství příkladů a indukci.
- **Učení se pozorováním a objevováním** (*learning from observations and discovery*)

Podoby informace o tom, že se systém učí správně

- Příklady, zařazené do tříd (konceptů), které se má systém naučit (učení s učitelem, *supervised learning*)
- Odměny za správné chování a tresty za nesprávné chování (posilované učení, *reinforcement learning*)
- Nepřímé náznaky (učení se napodobováním, *apprenticeship learning*)
- Žádná doplňková informace (učení bez učitele, *unsupervised learning*)

Metody strojového učení

- Prohledávání prostoru verzí (*version space search*)
- Techniky rozhodovacích stromů (*decision tree learning*)
- Techniky rozhodovacích pravidel (*learning sets of rules*)
- Techniky konceptuálního shlukování (*conceptual clustering techniques*)
- Posilované učení (*reinforcement learning*)
- Učení založené na vysvětlování (*explanation based learning*)
- Usuzování na základě analogií (*analogical reasoning*)
- Bayesovské učení (*Bayesian learning*)
- Fuzzy učení (*fuzzy learning*)
- Evoluční techniky (*evolution techniques*)
- Učení neuronových sítí (*artificial neural networks*)

Prohledávání prostoru verzí

Prostor verzí (version space)

Množina všech popisů konceptů konzistentních s trénovacími daty.

Algoritmy prohledávání prostoru verzí (Mitchell)

- Algoritmus generalizace (*specific to general search*)
- Algoritmus specializace (*general to specific search*)
- Algoritmus eliminace kandidátů (*candidate elimination algorithm*). Tento algoritmus udržuje dvě množiny kandidujících konceptů. *G* je množina maximálně obecných konceptů, *S* je množina maximálně specifických konceptů. Algoritmus specializuje *G* a generalizuje *S*.

Uvedené algoritmy používají jak pozitivní, tak i negativní příklady cílového konceptu. Je sice možné zobecňovat pouze z pozitivních příkladů, ale negativní příklady jsou důležité v prevenci přehnaného zobecnění.

Algoritmus eliminace kandidátů

1. Vlož do *G* nejobecnější koncept a do *S* prvý pozitivní příklad.
2. Pro každý pozitivní příklad *p* :
   2.1. Odstraň z *G* všechny prvky, které nepokrývají *p* .
   2.2. Každý prvek z *S* , který nepokrývá *p* , nahraď jeho nejmenší generalizací, která pokrývá *p* .
   2.3. Odstraň z *S* všechny prvky, které jsou obecnější než nějaké jiné prvky z *S* nebo obecnější než nějaké prvky z *G* .
3. Pro každý negativní příklad *n* :
   3.1. Odstraň z *S* všechny prvky, které pokrývají *n* .
   3.2. Každý prvek z *G* , který pokrývá *n* , nahraď jeho nejmenší specializací, která nepokrývá *n* .
   3.3. Odstraň z *G* všechny prvky, které jsou specifičtější než nějaké jiné prvky v *G* nebo specifičtější než nějaké prvky v *S* .
4. Jsou-li *G* a *S* prázdné, postup končí (neexistuje žádný koncept, který by pokrýval všechny pozitivní příklady a žádný negativní). Je-li *G* = *S* a obě množiny jsou jednoprvkové, postup rovněž končí (cílový koncept byl nalezen). V ostatních případech se vrať na krok 2.

Techniky rozhodovacích stromů

Vycházejí z příkladů popsaných vektory atributů a indexy tříd. Vytváří se rozhodovací strom, jehož nelistové uzly odpovídají atributům a listové uzly jsou ohodnoceny indexy tříd (každému atributu i každé třídě může odpovídat několik uzlů). Hrany vycházející z nelistového uzlu jsou ohodnoceny hodnotami příslušného atributu.

Základní algoritmy:

- ID3 (*Iterative Dichotomizer*) – Quinlan
- TDIDT (*Top-Down Induction of Decision Trees*)

Rozšířením algoritmu ID3 je systém C4.5.

Rozhodovací strom převedeme na soubor pravidel tak, že každé cestě z kořenového do listového uzlu odpovídá jedno pravidlo.

Vytvoření rozhodovacího stromu (ID3)

1. Vybere se jeden atribut jako kořen stromu.
2. Množina trénovacích příkladů se rozdělí na podmnožiny podle hodnot tohoto atributu.
3. Postupně se zpracují všechny tyto podmnožiny takto:
   a) Obsahuje-li podmnožina pouze příklady z téže třídy, vytvoří se pro tuto podmnožinu listový uzel a ohodnotí se indexem příslušné třídy.
   b) V opačném případě se vybere další atribut jako atribut podstromu. Tento atribut se pak zpracuje podle bodů 2 a 3.

Jako nejvhodnější může být vybrán např. atribut, který má

- nejmenší *entropii*
- největší *informační zisk* nebo největší *poměrný informační zisk*

Entropie atributu

Nechť A je atribut, $a_1, a_2, \ldots, a_n$ jsou jeho hodnoty a $c_1, c_2, \ldots, c_m$ jsou klasifikační třídy. Pak entropie atributu A se vypočte podle vztahů

$$H(A) = \sum_{j=1}^{n} P(a_j) H(a_j)$$

$$H(a_j) = -\sum_{i=1}^{m} P(c_i \mid a_j) \log\left(P(c_i \mid a_j)\right)$$

kde P znamená pravděpodobnost, logaritmus má základ větší než 1 (obvykle je základ roven 2) a pokud je pravděpodobnost rovna nule, pak se hodnota výrazu $P * \log(P)$ považuje za nulovou.

Informační zisk

Informační zisk je míra odvozená z entropie:

$$Zisk(A) = H(C) - H(A),$$

kde

$$H(C) = -\sum_{i=1}^{m} P(c_i)\log\left(P(c_i)\right) .$$

Nevýhodou entropie a informačního zisku může být skutečnost, že neberou v úvahu počet hodnot atributu. Proto se někdy používá *poměrný informační zisk*:

$$PoměrnýZisk(A) = \frac{Zisk(A)}{Větvení(A)} ,$$

kde

$$Větvení(A) = -\sum_{j=1}^{n} P(a_j)\log\left(P(a_j)\right)$$

Prořezávání rozhodovacích stromů

Možné problémy rozhodovacích stromů:

- Příliš velký rozsah stromu snižující jeho srozumitelnost.
- „Přeučení“ (*overfitting*) – dosažení neúměrné přesnosti stromu v situaci, že trénovací data jsou zatížena šumem.

Strom je možno zjednodušit tak, že místo toho, aby listovému uzlu odpovídaly pouze příklady jedné třídy, se spokojíme s tím, že příklady jedné třídy budou v listovém uzlu převažovat.

Zjednodušení (redukci) stromu je možno provést dvěma způsoby:

- Původní algoritmus se modifikuje doplněním nějakého kritéria, které indikuje, zda má uzel dále expandovat. Tímto způsobem se redukovaný strom vytvoří přímo.
- Vytvoří se úplný strom a následně se provede jeho prořezání (*post-prunning*). Postupuje se zdola nahoru a u každého podstromu se podle nějakého kritéria rozhoduje, zda se má podstrom nahradit listovým uzlem. Tento způsob je obvyklejší.

Zdokonalení technik rozhodovacích stromů

Práce s numerickými atributy:

V případě spojitých hodnot nebo velkého počtu diskrétních hodnot se obor hodnot rozdělí na intervaly, které se pak považují za diskrétní hodnoty atributu.

Chybějící hodnoty atributů:

Jednou z možností je uvažovat místo chybějící hodnoty nejčastější hodnotu příslušného atributu. Jinou možností je uvažovat všechny hodnoty atributu s vahami danými relativními četnostmi jejich výskytu v trénovacích datech.

Ceny atributů:

V některých aplikacích může hrát roli i cena zjištění hodnoty atributu. V takovém případě můžeme při tvorbě rozhodovacího stromu např. použít modifikované kritérium informačního zisku:

$$ZC(A) = \frac{Zisk(A)^2}{Cena(A)}$$

Techniky rozhodovacích pravidel

Odvození množin pravidel z rozhodovacích stromů

Algoritmy pokrývání množin:

- algoritmus AQ (Michalski)
- algoritmy CN2 (Clark, Niblet), CN4 (Bruha, Kočková)

U výše uvedených technik vytvářená pravidla neobsahují proměnné. Naproti tomu v *induktivním logickém programování* pravidla mohou obsahovat proměnné a mají tvar Hornových klauzulí 1. řádu (prologovských klauzulí).

Systémy induktivního logického programování:

- FOIL (Quinlan)
- PROGOL (Muggleton)
- TILDE (Blockeel, De Raedt)

Algoritmus AQ

Algoritmus AQ je určen rovněž pro učení z klasifikovaných příkladů popsaných pomocí atributů. Jeho výstupem je soubor pravidel popisujících všechny pozitivní příklady z trénovací množiny.

Algoritmus lze shrnout do následujících bodů:

1. Rozděl množinu příkladů na dvě podmnožiny: podmnožinu *P* pozitivních příkladů a podmnožinu *N* negativních příkladů.
2. Vyber z množiny *P* jeden příklad a označ jej *s* (*seed* neboli *jádro*).
3. Nalezni všechny maximální generalizace popisu jádra *s* tak, že jimi nesmí být pokryt žádný negativní příklad.
4. Podle vhodného preferenčního kritéria vyber nejlepší z těchto popisů a zařaď jej do množiny popisů. Odstraň z množiny *P* všechny příklady pokryté tímto popisem.
5. Je-li množina *P* prázdná, ukonči práci (výsledným popisem je disjunkce všech nalezených popisů). V opačném případě se vrať na bod 2.

Poznámky k algoritmu AQ

Případ více tříd:

Základní podoba algoritmu uvažuje dvě třídy. Rozšíření se obvykle provede tak, že pro každou třídu c_i se za pozitivní považují příklady této třídy a za negativní všechny ostatní. Jinou možností je vytvářet pravidla ke všem třídám současně (tak pracují algoritmy CN2, CN4).

Data zatížená šumem:

Algoritmus můžeme upravit tak, že v kroku 3 nepožadujeme, aby pravidlo pokrývalo příklady pouze jedné třídy.

Rozhodovací seznam:

Algoritmus AQ vytváří tzv. neuspořádaný soubor pravidel. Opakem je uspořádaný soubor pravidel neboli rozhodovací seznam (*decision list*), ve kterém jsou pravidla propojena pomocí ELSE. Podmínka za ELSE IF tedy implicitně obsahuje negaci podmínek všech předcházejících pravidel. Algoritmy CN2 a CN4 umožňují tvorbu uspořádaného i neuspořádaného souboru pravidel.

Algoritmus konceptuálního shlukování

1. Z množiny všech příkladů vyber k jader $s_1, \ldots, s_k$ (hodnotu k zadává uživatel). Výběr může být proveden náhodně nebo pomocí nějaké selekční funkce.
2. Pro každé jádro vytvoř maximálně obecný popis, který by je odlišil od všech ostatních jader. Každý takový popis je nazýván protokonceptem. Vytvořené protokoncepty nemusejí být disjunktní.
3. Uprav protokoncepty tak, aby byly disjunktní. Zpravidla existuje několik různých řešení (množin konceptů), z nichž se pomocí vyhodnocovací funkce vybere to nejlepší.
4. Není-li splněno zvolené kritérium ukončení, vyber nová jádra (z každého stávajícího protokonceptu se vybere jedno) a přejdi zpět ke kroku 2. V opačném případě vyber nejlépe ohodnocené řešení a proces ukonči.

Příklad tvorby disjunktních konceptů

1. Vytvoří se seznam L příkladů pokrytých více protokoncepty.
2. Jako počáteční hodnoty vytvářených disjunktních konceptů C_i se položí $C_i = \{s_i\}$, (i = 1, … , k).
3. Vezme se první příklad ze seznamu L a odstraní se z tohoto seznamu.
4. Pro každý z protokonceptů P_i pokrývajících vybraný příklad e se pokusně vytvoří nový, co nejobecnější popis tak, aby pokrýval příklad e a shluk C_i a nepřekrýval se s žádným z ostatních konceptů.
5. Pomocí ohodnocovací funkce vybereme nejlepší z těchto popisů. Nechť je to popis odpovídající protokonceptu P_j. Příklad e přidáme ke shluku C_j . Je-li seznam L prázdný, postup končí. V opačném případě se vracíme na krok 3.

Poznámky ke konceptuálnímu shlukování

Vyhodnocovací funkce

Může zahrnovat řadu kritérií, jako např. míru rozdílnosti mezi shluky, počet proměnných umožňujících diskriminovat mezi shluky, jednoduchost popisu shluků, apod.

Ukončovací kritérium

Může to být např. počet iterací, po které nedošlo k významnému zlepšení vyhodnocovací funkce.

Volba jader ze stávajících shluků

V algoritmu CLUSTER/2 (Michalski) se s využitím vhodné metriky vybírá prvek nejbližší středu každého shluku. V případě neuspokojivých shluků, které se po několik iterací nezlepšují, se jako nová jádra vybírají prvky nejbližší k hranici shluku.

Učení založené na vysvětlování

Složky učení založeného na vysvětlování:

- cílový koncept, jehož definici je třeba vytvořit,
- trénovací příklad jako instance cíle,
- doménová teorie (množina pravidel a faktů, které jsou používány k vysvětlení, proč je trénovací příklad instancí cílového konceptu),
- kritéria popisující tvar, který definice cíle mohou nabývat.

V prvém kroku se pomocí doménové teorie konstruuje vysvětlení trénovacího příkladu. Obvykle se jedná o důkaz, že příklad logicky vyplývá z teorie, přičemž tento důkaz může být reprezentován např. pomocí stromové struktury.

V dalším kroku je zobecněním tohoto vysvětlení získána definice cílového konceptu. Zobecnění spočívá v substituci proměnných za ty konstanty v důkazním stromu, které závisejí pouze na trénovacím příkladu. Na základě zobecněného důkazního stromu se definuje pravidlo, jehož závěr je kořen stromu a předpoklad je konjunkcí listů.

Konstrukce zobecněného důkazního stromu

Existuje řada způsobů konstrukce zobecněného důkazního stromu. Jednou z možností je (Mitchell et al.) nejprve zkonstruovat důkazní strom, který je specifický pro trénovací příklad, a následně tento důkaz zobecnit tzv. cílovou regresí (*goal regression*). Cílová regrese unifikuje zobecněný cíl s kořenem důkazního stromu pomocí potřebných substitucí. Algoritmus rekurzívně aplikuje tyto substituce na další uzly stromu, dokud všechny vhodné konstanty nejsou zobecněny.

DeJong and Mooney navrhují alternativní přístup, který v podstatě spočívá v paralelním vytváření zobecněného i specifického stromu. To je uskutečněno pomocí tzv. vysvětlovací struktury (*explanation structure*), která reprezentuje abstraktní strukturu důkazu. Při vytváření této struktury se konstruují dva substituční seznamy, jeden pro specifický strom a druhý pro obecný strom.

Metody založené na analogii

Učení nových konceptů nebo odvozování nových řešení prostřednictvím použití podobných konceptů a jejich řešení.

Metody:

- Případové usuzování (*Case-Based Reasoning*, CBR)
- Pravidlo nejbližšího souseda (*nearest neigbour rule*)
- Učení založené na instancích (*Instance-Based Learning*, IBL)
- Líné učení (*lazy learning*)
- Paměťové učení (*Memory-Based Learning*)

Znalosti jsou reprezentovány v podobě báze již vyřešených problémů. Při usuzování se k danému problému v této bázi hledá nejpodobnější případ a jeho řešení se adaptuje na novou situaci.

Fáze učení pak představuje zapamatování znalosti z vyřešeného problému pro budoucí použití. Při učení se tedy neprovádí generalizace z příkladů (proto název *líné učení*).

Případové usuzování

CBR je založeno na vyhledávání a přizpůsobování starých řešení novým problémům. Obecný CBR cyklus lze popsat pomocí následujících čtyř kroků:

1. Vyhledání (*retrieving*) nejvíce podobného případu nebo případů.
2. Použití (*reusing*) informací a znalostí z tohoto případu k vyřešení daného problému.
3. Revize (*revising*) navrženého řešení.
4. Uchování (*retaining*) částí těchto zkušeností tak, aby byly použitelné pro řešení budoucích problémů.

Kromě specifických znalostí reprezentovaných případy hrají určitou roli v tomto procesu i obecné znalosti z příslušného oboru.

Bayesovské učení

Bayesovské uvažování

- je základem pro algoritmy učení, které přímo manipulují s pravděpodobnostmi
- je nástrojem pro analýzu a pochopení ostatních algoritmů strojového učení

Výhodou metod bayesovského učení je schopnost klasifikovat příklady do tříd s určitou pravděpodobností.

Metody Bayesovského učení:

- Bayesovský optimální klasifikátor
- Gibbsův algoritmus
- Naivní bayesovský klasifikátor
- Učení bayesovských sítí

Naivní bayesovský klasifikátor

Naivní bayesovský klasifikátor vychází z předpokladu, že jednotlivé evidence E_1, … E_K jsou podmíněně nezávislé při platnosti hypotézy H. Pak platí

$$P(H \mid E_1,\ldots,E_K) = \frac{P(H)}{P(E_1,\ldots,E_K)} \prod_{i=1}^{K} P(E_i \mid H)$$

Naivní bayesovský klasifikátor:

$$H_{NB} = \arg\max_{H_j} P(H_j) \prod_{i=1}^{K} P(E_i \mid H_j)$$

Vysvětlení pojmů:

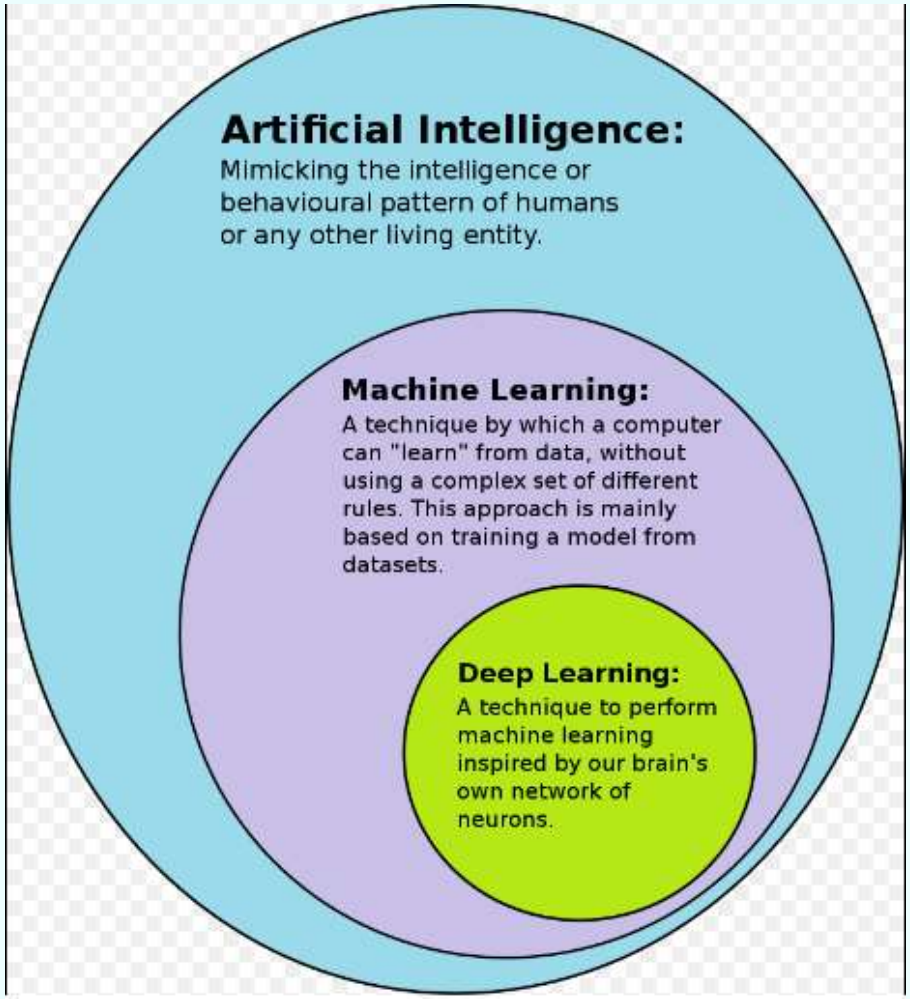
AQ-Algorithm – algorithm quasi-optimal

koncept – první předběžné zpracování, návrh, náčrt (zejména textu); nákres, osnova, skica, rozvrh

atribut – neodlučitelná vlastnost, podstatný znak, přívlastek; dále atributy jsou speciální části v kódu psaného v jazycích HTML nebo XHTML; atributy jsou jakési parametry, které umožňují jednotlivým značkám přiřazovat speciální vlastnosti …

atribut (z lat. *ad-tribuere*, přidělovat, připisovat, přisuzovat) je dále rozlišovací rys, podstatná vlastnost nějakého objektu a v obecném užití charakteristická vlastnost, v gramatice přívlastek, symbol nebo emblém držitele úřadu (např. rektorské žezlo, královská koruna), v ikonografii rozlišovací znak významné postavy, např. světce (korouhev sv. Václava), a také dodatečná informace připojená k souboru v souborovém systému.

Hluboké učení (Deep Learning) – zhruba od roku 2010



Moderní technika z oblasti strojového učení inspirovaná biologickou neuronovou sítí člověka, založená na umělých neuronových sítích s reprezentačním učením. Učení může být pod dohledem (supervised learning), s částečným dohledem (semi-supervised learning) nebo bez dohledu (unsupervised learning).

Hluboké učení (pedagogická definice) je učení aktivní, při kterém studenti používají **kritické a kreativní myšlení**, **spolupracují** při řešení problémů, **učí se jeden od druhého** a poznatky jsou získávány **skrz prožitek**. Učitel žáky **motivuje a navádí správným směrem** a pomáhá jim **utvářet si cíle**. Hluboké učení je aktivní na rozdíl od učení povrchního, které je pasivní.

Povrchní učení má také zásadní roli a v žádném případě ho nemůžeme zatracovat. Povrchní učení znamená **získávání informací, faktů a dovedností**. Až pak může přijít na řadu **hluboké učení**, při kterém studenti **rozvíjejí nápady**, zjištují **podobnosti, odlišnosti a vzorce**, aplikují povrchně nabyté poznatky **v nových kontextech** a umí je **pospojovat s ostatními vědomostmi**.

Poznámka: 90% výuky pro začátečníky je a musí být povrchní učení, ale později bychom měli přepnout na hluboký program. Je tudíž nutné zvolit správné načasování a identifikovat, kdy už je načase přestat se drtit a memorovat fakta (či slovíčka) a posunout se dál.

Hluboké učení neboli **deep learning** v AI je disciplína v rámci strojového učení, která se zabývá využitím algoritmů (především neuronových sítí) s velkým počtem vrstev (layers) reprezentujících data. *Hloubkou modelu* se nazývá počet vrstev (obvykle složených z umělých neuronů), které jsou za sebou propojeny tak, že výstup jedné z nich je vstupem následující vrstvy. U modelů hlubokého učení se přitom hloubka nachází často v řádech desítek a více vrstev.

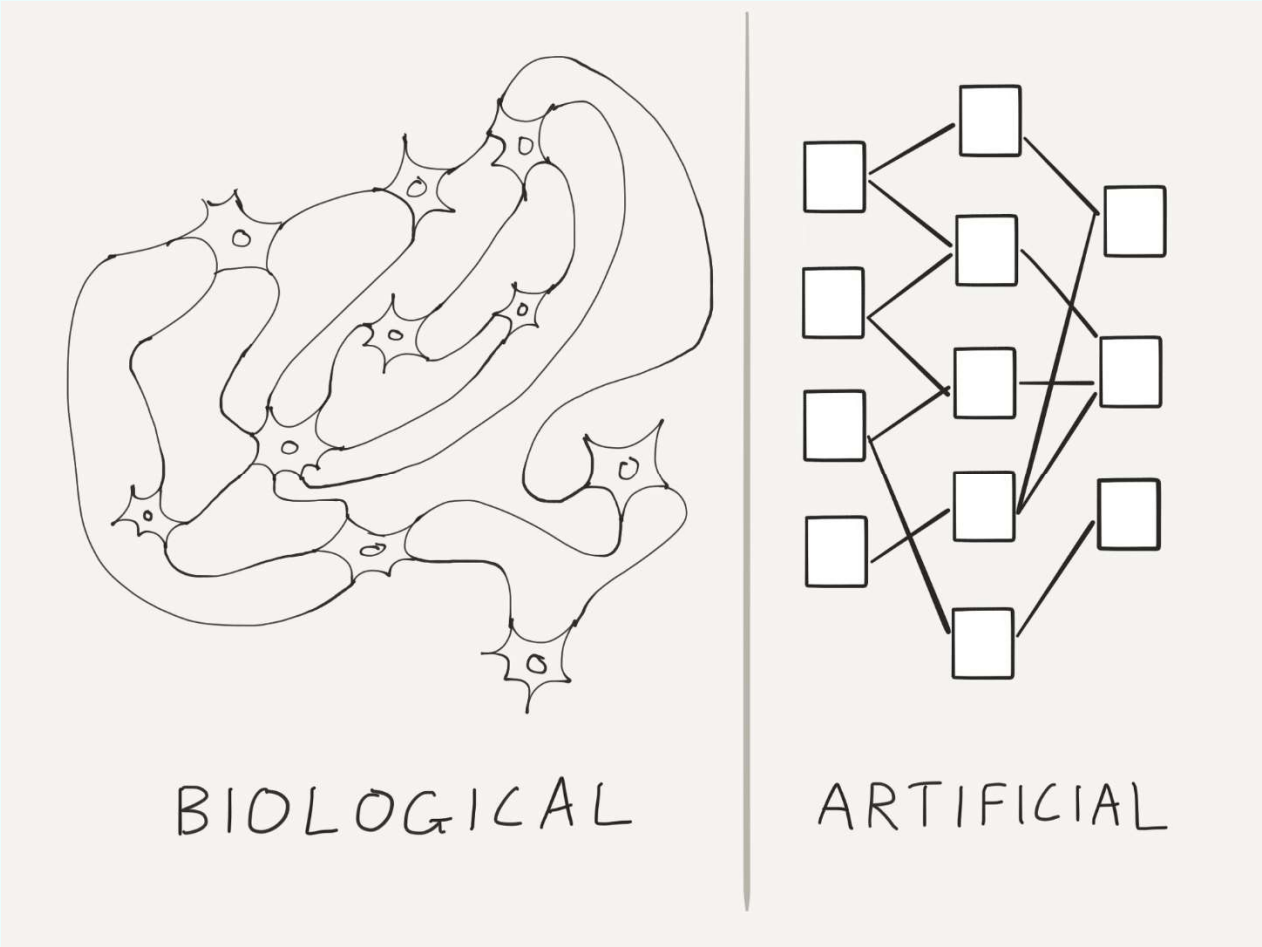
Pro odhad parametrů sítě (trénování) se obvykle používá algoritmus zpětného šíření chyby (backpropagation algorithm). Metodologie hlubokého učení se prosadila kolem roku 2010 jako základní možnost pro řešení složitých problémů strojového učení jako je klasifikace obrazů, promluv a videí nebo překlady z jednoho přirozeného jazyka do jiného.

Lidský mozek může být popsán jako biologická **neuronová síť** — jednotlivé neurony jsou vzájemně propojeny a navzájem si přeposílají signály a transformují je pomocí určitých vzorců. Neuron má libovolný počet vstupů, ale pouze jeden výstup.

Mozkem, respektive obecně nervovým systémem, jsou inspirovány **umělé neuronové sítě**. Podstatou umělých neuronových sítí je struktura systému pro zpracování informací složená z velkého počtu propojených elementů (neuronů), které společně pracují na řešení specifického problému. Stejně jako lidé se učí z příkladů, tak umělé neuronové sítě se během učení přizpůsobují změně a nastavují parametry pro specifické použití, jako je rozpoznání obrazů (vzorů) nebo klasifikace dat.

Běžné výpočetní systémy jsou programovány procedurálně — program začíná prvním řádkem kódu a po jeho vykonání následuje další, ale pravé neuronové sítě neběží lineárně, ale paralelně ve všech uzlech.

Příklad: Biologická a umělá neuronová síť



Učení s učitelem (používají např. Support Vector Machines – SVM) je jednou z nejvíce používaných forem strojového učení. Pokud je cílem vytvořit systém, který umožňuje určit, zda na obrázku je dům, auto nebo člověk, je nejprve potřeba připravit velkou tzv. trénovací množinu dat, která obsahuje obrázky domů, aut a lidí, kde každý jednotlivý obrázek obsahuje informaci, do které z kategorií (tříd) objekt spadá. Avšak taková množina dat s popisy se často připravuje ručně nebo jen částečně automaticky, což je hlavní nevýhoda učení s učitelem. Během tréninku pak systém prochází obrázky a vytvoří výstup ve formě vektoru hodnocení pro každou kategorii (třídu). Cílem je, aby všechny existující třídy měly co nejvyšší tzv. ohodnocující skóre, ale takový výsledek je bez trénování nepravděpodobný. Během procesu učení se proto počítá funkce, která měří chybu mezi hodnocením výstupu a vzorovým hodnocením. Následně systém upraví své nastavení s cílem snížení této chyby. Po trénování se výkon nastaveného systému ověří na jiném datovém balíku, který se nazývá testovací množina. Ta slouží k otestování obecných schopností nastaveného systému a určení výkonu na datech, se kterými se systém ještě nesetkal.

Architektury sítí pro hluboké učení, jako jsou hluboké neuronové sítě, sítě hlubokého učení, opakující se neuronové sítě a konvoluční neuronové sítě, byly použity v oblastech jako je počítačové vidění, strojové vidění, rozpoznávání řeči, zpracování přirozeného jazyka, rozpoznávání zvuků, filtrování sociálních sítí, strojový překlad, bioinformatika, analýza medicínských obrazů, kontrola materiálů a programy deskových her, kde přinesly výsledky zcela srovnatelné s výkony lidských expertů a v některých případech i lepší.

Poznámka: Přívlastek „hluboký“ v hlubokém učení pochází z použití více vrstev ve struktuře sítě. Rané práce ukázaly, že lineární perceptron nemůže být univerzálním klasifikátorem, ale může to být např. síť s nepolynomiální aktivační funkcí s jednou skrytou vrstvou neomezené šířky. Hluboké učení sítí je moderní variace struktur neuronových sítí, která se týká neomezeného počtu vrstev omezené velikosti, což umožňuje praktickou aplikaci a optimalizovanou implementaci při zachování teoretické univerzality za definovaných podmínek. Při hlubokém učení je také povoleno, aby byly vrstvy heterogenní a aby se z důvodu efektivity, trénovatelnosti a srozumitelnosti sítě tyto značně odchýlily od struktur biologických neuronových sítí.

Hluboké neuronové sítě jsou obecně interpretovány ve smyslu věty o univerzální aproximaci (universal approximation theorem) nebo věty o pravděpodobnostní inferenci.

Klasická věta o univerzální aproximaci se týká kapacity dopředných neuronových sítí s jedinou skrytou vrstvou konečné velikosti k aproximaci spojitých funkcí. První důkaz byl publikován v roce 1989 George Cybenkem pro „esovité“ aktivační funkce a byl zobecněn na architektury vícevrstvých sítí v roce 1991 Kurtem Horníkem. Nedávná práce také ukázala, že univerzální aproximace platí také pro neomezené aktivační funkce, jako je např. rektifikovaná lineární jednotka (rectified linear unit).

Věta o univerzální aproximaci pro hluboké neurální sítě se týká kapacity sítí s omezenou šířkou, ale hloubka reprezentace se může zvětšovat. Pokud je šířka hluboké neuronové sítě s tzv. ReLU aktivací striktně větší než vstupní dimenze, pak síť může aproximovat jakoukoli Lebesgueovu integrovatelnou funkci; pokud je šířka menší nebo rovná vstupní dimenzi, pak hluboká neuronová síť není univerzálním aproximátorem.

Pravděpodobnostní interpretace je odvozena z metod strojového učení. Je založena na inferenci a na optimalizaci trénování a testování systému za účelem zobecnění možností dosažení optimální funkce sítě. Pravděpodobnostní interpretace používá za aktivační nelinearitu sítě kumulativní distribuční funkci, která ale někdy může vést k „pádu“ funkce sítě.

Pravděpodobnostní interpretace byla vyvinuta mnoha výzkumníky, včetně Hopfielda, Widrowa a Narendry, ale poprvé byla srozumitelně publikována v knize Ch. M. Bishop: Neural Networks for Pattern Recognition.

Integrace hlubokého učení s internetem věcí

Hluboké učení je všudypřítomné, aniž si to mnohdy uvědomujeme. Tato forma umělé inteligence řídí vaše sociální média a pomáhá vám při hledání na Google. A možná nepotrvá dlouho a hluboké učení bude monitorovat vaše životně důležité orgány nebo ovládat spínání termostatu.

Výzkumníci z Massachusettského technologického institutu (MIT) např. vyvinuli systém, který umožní integraci neuronových sítí hlubokého učení s novými a mnohem menšími zařízeními, jako jsou drobné počítačové čipy v nositelných lékařských zařízeních, domácích spotřebičích a 250 miliardách dalších zařízení, které společně tvoří internet věcí.

Nový systém, nazvaný MCUNet, navrhuje kompaktní neuronové sítě poskytující nebývalou rychlost a přesnost pro integraci hlubokého učení s fyzickými zařízeními vybavenými elektronikou, softwarem, senzory, pohyblivými částmi a síťovou konektivitou, která umožňuje těmto zařízením se propojit a vyměňovat data. Technologie by mohla v budoucnu usnadnit rozšíření internetu věcí a současně zlepšit bezpečnost přenášených dat.

Závěr – doporučené zdroje informací:

- zřejmě nejrozsáhlejší ucelený přehled oblasti Deep Learning: Learning Deep Architectures for AI
- jednoduchý úvod do problematiky: An Introduction to Deep Learning
- online kurz na Coursera.org od Geoffrey E. Hintona: Neural Networks for Machine Learning
- přednáška Geoffrey E. Hintona na Youtube: Recent Developments in Deep Learning
- úvod do problematiky autoenkodérů: UFLDL Tutorial
- úvod do problematiky Restricted Boltzmann Machines: Introduction to Restricted Boltzmann Machines
- celý web věnovaný problematice včetně knihovny Thenano pro jednoduché programování neuronových sítí v Pythonu: http://deeplearning.net
- video tutoriál k využití Deep Learning v oblasti počítačového zpracování přirozeného jazyka: Deep Learning for NLP

Významná jména z oblasti strojového učení:

Tom Mitchell – University Professor (Tom.Mitchell@cmu.edu)
Machine Learning Department (chair)
School of Computer Science
Carnegie Mellon University, Pittsburgh

John Ross Quinlan – is a computer science researcher in data mining and decision theory. He has contributed extensively to the development of decision tree algorithms, including inventing the canonical C4.5 and ID3 algorithms. He is currently running the company RuleQuest Research which he founded in 1997.