AI background

Worksheet for **mentors**



Harmonogram:

Pozn.: časové rozvržení není zaznamenáno. Je pouze obsah.

Obsah:

* N-dimenzionální prostředí
  + Cílem bylo popsat jak vypadají data, která mají hodně parametrů a vizuálně je přiblížit
  + Colab **Počet\_kategorii.ipynb** (bylo by fajn tam přidat načítání z csv)
* Derivace v prostoru
  + Gradient a hledání lokálního minima, přiblížení hledání minima k učení neuronových sítí <https://youtu.be/IHZwWFHWa-w>
  + Bohužel interaktivní colab není vytvořen inspirace zde <https://youtu.be/2-Ol7ZB0MmU>
  + Fungování konvolučních vrstev <https://youtu.be/YRhxdVk_sIs>
* Operace s vektory
  + Přiblížení toho, že vstupem do neuronových sítí je vektor
  + Operace s vektory (násobení, sčítání) – ideálně připravit nějakou grafiku, která to bude vysvětlovat
* Fotka + zvuk na vektor
  + Přibližný popis převodu….převod fotky lze ukázat na **Zpracovani\_obrazku.ipynb**
  + Existuje více principů převodu na vektor
* Projití colabu **Co\_dela\_AI.ipynb** 
  + Představení algoritmického řešení úloh nastavených na předem určené hodnoty a srovnání s AI
* Seznámení s fungováním neuronové sítě na příkladu
  + Inspirace v playlistu 3blue 1brown <https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>
* Tensorflow
  + Seznámení s Tensorflow a jeho představení, úvod atd.
* Závěr
  + Proč se používají grafické karty + nějaká ukázka

Nvias, z. s.

Cukrovarská 20

30100 Plzeň

Další materiál pro studium CNN: <http://neuralnetworksanddeeplearning.com/>

<http://colah.github.io/>

Text ke slidům:

NEURONOVÁ SÍŤ

* V základu jsou inspirovány lidským mozkem. Motivací bylo a je strojově napodobit chování geniálního lidského mozku strojem. Ve značně limitovaných případech se daří lidský mozek i překonat (sledování patternů ve značně zašuměných nebo nepřehledných datech)
* Skládají se z jednotlivých neuronů uspořádaných do oněch NN. Jeden neuron má schopnost udržet určitou hodnotu v daném rozsahu activation funkce, což lze přirovnat k pravděpodobnosti výsledku (jak si je s výsledkem NN jistá)
* Je více typů neuronových sítí (NN) například CNN (Convolutional Neural Network), která se používá pro rozpoznání obrazu nebo RNN (Recurrent Neural Network) pro rozpoznání zvuku
* Pro zjednodušení se budeme věnovat pouze jednoduché struktuře MLP…ostatní typy fungují na podobném principu a proto je zbytečné hned začínat se složitějšími typy hned na začátku
* OBRÁZEK!!! Na obrázku můžeme vidět jednoduchou NN se strukturou tzv MultiLayer Perceptron (MLP) neboli Vanilla NN. Nalevo je vstupní část NN, někdy nazývaná jako activation layer, uprostřed jsou tzv. skryté vrstvy (hidden layers) a vpravo lze vidět výstupní vrstvu (output layer).

DATA

* Jsou důležitou komponentou většiny úloh spojených s AI. Problematiku sběru dat a to jak by měla vypadat si probereme v některém z našich dalších workshopů zaměřených už přímo na danou tematiku.
* Nyní si pouze odpovíme na otázku, jak vlastně vypadají data, která NN dostane na zpracování. Pokud máme k dispozici data například hmotností a typů dopravních prostředků, máme jednoduchou 2D tabulku, kde na jedné ose budou informace o počtu zákazníků, kteří navštívili stránku, na druhé bude, z jaké platformy byli přesměrováni . Pokud se však rozhodneme zpracovávat komplexnější data s více informacemi, je potřeba rozšířit naši tabulku o další osu neboli dimenzi. Například jak dlouho daný zákazník z dané platformy stráví na našem webu, co je nejvíce zajímá, jaký je jejich věk, pohlaví, poloha, jazyk, typ reklamy, jak daleko se proklikají webem, čas kdy web navštívili, čas zveřejnění reklamy nebo příspěvku na dané téma.
* Pro lepší porozumění si otevřete colab notebook na odkaze …. Zbytek si vysvětlíme tam,

Konstrukce NN

* Konstrukce NN je nejednoznačná, protože neexistuje jednoznačně definovaný nejlepší způsob jakou strukturu zvolit. Je to převážně na bázi testování a zkoušení, zdali to bude fungovat lépe než jiný případ nebo ne.
* Vhodnou konstrukci NN si můžeme vyzkoušet na <https://playground.tensorflow.org/>
* To jak skryté vrstvy budou zpracovávat vstupní data je dáno několika parametry a funkcemi
  + Weights – váhy jednotlivých vazeb neuronů
  + Activation – hodnota uložená v neuronu
  + Bias – zesílení neboli threshold, kterým se nastaví citlivost
* Ta funkce, která určuje výslednou hodnotu následujícího neuronu se nazývá activation function a je jich více typů…více na dalším slidu….nejdříve ale…

Jak funguje NN

* Z funkce nahoře lze vidět, jakým způsobem se vypočítává hodnota následujícího neuronu. Výslednou hodnotu získáme z tzv activation funkce, která si jako parametry bere hodnotu aktivace předcházejících neuronů vynásobené váhami (weight) a sečtenou s thresholdem zvaným bias(jeden pro celý součet součinů).
* Toto probíhá pro všechny následující neurony

Co to je „activation function“?

* Aby nedocházelo ke kumulaci vysokých hodnot v jednom neuronu (a urychlilo se učení), používají se pro výpočet hodnoty neuronu různé aktivační funkce, které omezí rozsah oněch hodnot.
* Fungují na principu, že zvyšují rozdíly mezi kladnými a zápornými hodnotami (transformují dané hodnoty na daný omezený interval)
* X je v následujících rovnicích vážená suna předchozích neuronů navázaných na jeden v následující vrstvě
  + Sigmoid
  + ReLU
  + Tanh
  + Softmax

NN-vstupy

* Vstupem do NN je vektor. Při zpracování trénovacích dat je transformujeme do vektoru ať je to obrázek (obrázek se jako matice pixelů po řádcích načte do jednoho dlouhého vektoru), zvuk (Fourierova transformace a její další zpracování), či jsou to jiná data ve formě vektorů.
* Pro zjednodušení výpočtů se transformují do maticového počtu, kde weights jsou poskládány do matice a vektory activation a bias do vektorů
* Násobení matice vektorem lze znázornit i graficky. Matici vah (weights) můžeme brát jako transformační matici, které vykreslíme sloupcové vektory a odpovídající hodnotou activation je vynásobíme. V podstatě tak transformujeme vstupní hondoty lineární transformací do jiného lineárního prostoru o stejné dimenzi.

Trénování NN

* Pro trénování NN potřebujeme speciálně vytvořené datasety. Které obsahují jak surová nebo předzpracovaná data a tzv anotace. Jde o popis správného výsledku NN. Například v datasetu obrázků budeme mít fotku kočky a anotace bude „kočka“, pokud by byla fotografie byla se psem, bude v anotaci uveden „pes“. Při více kategoriích se používají klíče nebo indexy a k tomu odpovídající tabulka, kde lze najít podrobná popis klasifikovaného objektu.
* Před trénováním nastavíme veškeré vnitřní parametry (weights, biases) na náhodné hodnoty (v případě předtrénovaného modelu se nahrají již natrénované parametry). Obdoba seedu v minecraftu kde například hledáme diamanty.
* Začneme posílat trénovací data na jejichž základě počítáme cost funkci jako sumu druhých mocnin rozdílů výstupu NN a anotace daných dat neboli správného výsledku.
* Cost funkce je skvělá pro celkové hodnocení kvality NN. Jako parametry bere všechny weight a bias. A výstupem je jedna hodnota (chyba sítě)…s rostoucím počtem parametrů roste složitost a je možné si ji tedy představit pokud má NN pouze jeden nebo dva parametry…vznikne prostorová funkce a minimalizací můžeme chápat hledání lokálního minima na nějakém povrchu(cost funkci).

Trénování NN-minimum

* Proč hledáme pouze lokální minimum a ne globální minimum? Protože by to bylo značně obtížné a často nemožné, proto se budeme muset spokojit s určitou nedokonalostí v podobě lokálního minima
* Popis obrázků a demonstrace hledání minima při náhodném zvolení počátečních parametrů

Trénování NN-gradient

* Jak najdeme minimum ve více dimenzích? Odpovědí na tuto otázku jsou derivace, v našem případě parciální derivace. (VYGOOGLI SI TO) Konkrétně gradient descent nám dá odpověď na to jakým směrem jít, aby cost funkce klesala co nejrychleji. (vrací n-vektor) Hledání postupuje po jednotlivých krocích, kde každý krok znamená korekci potřebných parametrů, které nám poradí právě onen gradient. Zejména i velikost kroků (citlivost otáčení knoflíky) má velký vliv, jestli minimum najdeme dostatečně malé nebo ne. Zároveň také dostáváme informaci o tom, které parametry se nám nejvíce vyplatí měnit.

Trénování NN-backpropagation

* Jak nastavíme parametry? Backpropagation je odpovědí. Jedná se o zpětné nastavení parametrů na základě gradientu cost funkce.
* Alternativní popis backpropagation:
  + Popsat na základě obrázku
  + Nastavení parametrů probíhá od výstupu NN směrem ke vstupu. Na základě rozdílů ve výsledku NN a správné odpovědi proporcionálně upravíme weights and biases. Každý výstupní neuron bude chtít upravit individuální parametry, ze kterých byl vypočten. Změny ze všech výstupních neuronů se v předchozích neuronech sečtou a opět upravíme individuální parametry a propagujeme dále až ke vstupní vrstvě.
  + Pro každý jeden vzorek trénovacích by se teoreticky měla tato inverzní propagace (korekce) provádět. Když však veškeré požadované změny parametrů zprůměrujeme….dostaneme tzv negativní gradient naší cost funkce (nebo jeho ekvivalent lišící se nějakým násobkem)
* Průběh backpropagation má na svědomí ono učení NN, tedy nastavování parametrů a hledání lokálního minima.
* Ideální teoretický průběh by byl nechat pro každý vzorek z trénovací sady vypočíst korekci, zaznamenat a udělat průměrnou hodnotu změny individuálních parametrů (neboli gradient). Následně by se upravily parametry směrem nejprudšího klesání a process by se opakoval dokud by se lokální minimum nenalezlo.

Trénování NN -batches

* Protože propagace každého jednotlivého kroku gradientu trvá extrémně dlouhou dobu (pro výpočet přesného gradientu by se muselo zjišťovat korekce pro parametry u potencionálně mnoha tisícům nebo milionům trénovacích dat – pro jeden krok/korekci parametrů)
* Počítáme tedy s tzv. stochastickým gradientem, který je sumou dílčích gradientů jednotlivých batchů
* Před jednotlivým nastavením parametrů a korekcím nám stačí jen část trénovacích dat

Trénování -detail backpropagation

* Vysvětleno ve videu…shlédni <https://youtu.be/tIeHLnjs5U8>

Tensorflow

* Povídání na základě stránek a slidů
* Zmínit keras apod.

Grafické karty

* Zakončení
* Proč karty???
* Pustit video kde se porovná CPU a GPU
* GPU dokáže zpracovávat data více paralelně