Využití autoenkodéru k detekci odlehlých pozorování

Lukáš Hromadník

2019

Obsah

1 Využití autoenkodéru k detekci odlehlých pozorování

1.1 Úvod

Autoenkodér je poměrně specifická neuronová síť, která se skládá ze dvou částí - enkodéru a dekodéru. Enkodér vytváří zakódovanou reprezentaci vstupních dat a dekodér se snaží tuto zakódovanou reprezentaci převést zpět do původní podoby. Tento problém lze nazvat jako redukce dimenzionality.

Podobným problém se zabývá Analýza hlavních komponent (PCA). Ta se také používá ke zmenšení dimenze vstupních dat s co nejmenší ztrátou informace. PCA však k mapování dat do nového menšího prostoru používá pouze lineární transformace, kdežto autoenkódér může použít i nelineární.

Učení autoenkodéru probíhá formou učením bez učitele. To znamená, že jednotlivá vstupní data nemají k sobě přiřazené výsledné ohodnocení. Učení bez učitele je však v případě autoenkodéru trochu odlišné. Jednotlivá vstupní data jsou porovnávány s výstupem a síť samotná se pak během učení snaží upravit parametry tak, aby minimalizovala ztrátovou funkci, která zohledňuje rozdíl mezi vstupem a výstupem. Nejčastěji se jako ztrátová funkce používá střední kvadratická chyba.

1.2 Zadání

- Vyzkoušejte dvě různé architektury autoenkodéru jako např. undercomplete autoencoder, sparse autoencoder, denoising autoencoder. U obou architektur se zaměřte na hledání optimálních hodnot hyperparametrů.
- Vyberte si libovolný z datasetů na stránce http://odds.cs.stonybrook.edu/. Jedinou podmínkou je, aby měl více něž 20 příznaků a méně než 20 % odlehlých pozorování.
- Pokud je v datech informace o tom, že se jedná o odlehlé pozorování, využijte ji pouze k výsledné evaluaci, nikoliv pro trénování.
- Výsledky pro obě testované architektury porovnejte a diskutujte.

1.3 Řešení

Mezi některé architektury autoenkodérů patří například:

- undercomplete autoenkodér,
- sparse autoenkodér,
- denoising autoenkodér,
- convolutional autoenkodér,
- deep autoenkodér.

Pro řešení problému jsem se rozhodl použít architektury **undercomplete autoenkodér** a **sparse autoenkodér**.

Chceme-li autoenkodér využít k detekci odlehlých pozorování, je vhodné ho naučit pouze na datech, o kterých víme, že nepatří mezi odlehlá pozorování. Díky tomu se autoenkodér více přiblíží běžným pozorováním a nebude nucen upravovat parametry sítě tak, aby se během učení minimalizovala ztrátová funkce i pro odlehlá pozorování.

Undercomplete autoenkodér je jednoduchá neuronová síť, která má jednu skrytou vrstvu obsahující menší počet neuronů než je velikost vstupu. Díky tomu se vytvoři "bottleneck" a autoenkodér je nucen naučit se z dat pouze to nejnutnější.

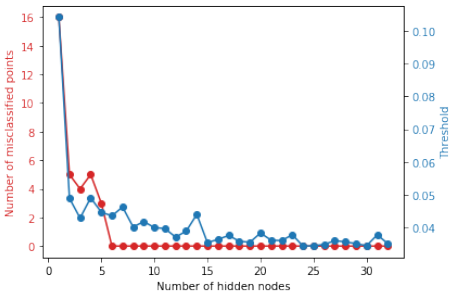
Velice podobnou architekturu má i deep autoenkodér. Ten se však liší tím, že počet vnitřních skrytých vrstev je vyšší než 1.

Sparse autoenkodér je opět jednuchá neuronová síť. Opět obsahuje jednu skrytou vrstvu, která může mít libovolný počet neuronů. Rozdíl proti undercomplete autoenkodéru je v tom, že ztrátová funkce má navíc penalizační část, která v sobě navíc kombinuje hodnoty aktivačních funkcí neuronů ve skryté vrstvě. Minimalizací takto nastavené ztrátové funkce dojde k tomu, že během učení jsou některé neurony "vypnuty". Autoenkodér je tedy opět donucen k tomu, aby nepoužíval všechny dostupné informace ale pouze ty nejdůležitější.

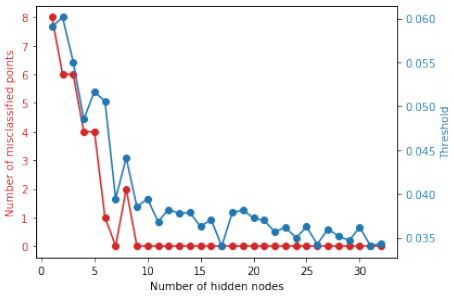
V implementační části jsem vytvořil dva autoenkodéry - undercomplete a sparse. Oba byly natrénovány na datasetu Letter recognition, který obsahoval 1600 datových bodů s dimenzí 32. Počet odlehlých pozorování byl 100, což je 6.25 % z celkového počtu.

2 Výsledky

V první části jsem se zaměřil na hledání optimálního počtu neuronů ve skryté vrstvě. Obě architektury jsem učil postupně s jedním, dvěma, třemi až 32 neurony (32 je dimenze datasetu) a zaznamenával výsledky.



Obrázek 1: Undercomplete autoenkodér



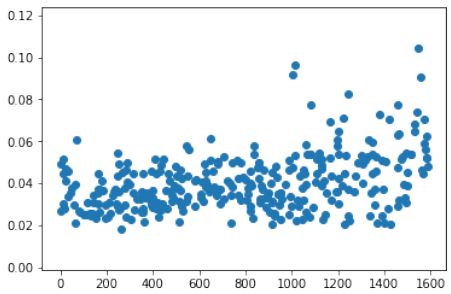
Obrázek 2: Sparse autoenkodér

Z grafů je možné vidět, že již se šesti neurony ve skryté vrstvě byly oba autoenkodéry schopné s nulovou chybou od sebe oddělit běžná pozorování a odlehlá pozorování.

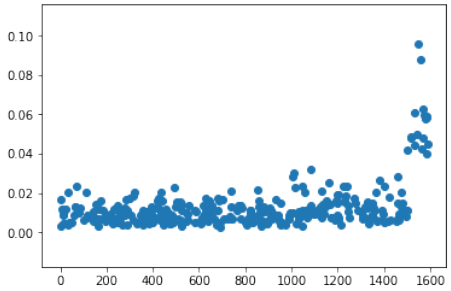
Také je z grafů jasně vidět, jak se zvětšujícím se počtem neuronů ve skryté vrstvě stoupala úspěšnost. Ve chvíli počet neuronů překročil hranici 6, oba autoenkodéry byly schopné klasifikovat pozorování s prakticky nulovou chybou.

Současně s počtem špatně klasifikovaných pozorování je v grafech znázorněna i hranice, která byla zvolena jako nejvhodnější pro klasifikaci. Této hranice bylo dosaženo tak, že jsem postupně procházel ohodnocení odlehlých pozorování a pro každou z těchto hodnot jsem provedl klasifikaci celého testovacího datasetu. Klasifikace probíhala tak, že pokud ohodnocení daného bodu bylo nad touto hranicí, byl označen za odlehlé pozorování, a pokud ohodnocení bylo pod touto hranicí, byl označen za běžné pozorování. Takto jsem proiteroval přes všechny odlehlá pozorování a vybral hranici, která vracela nejlepší výsledky.

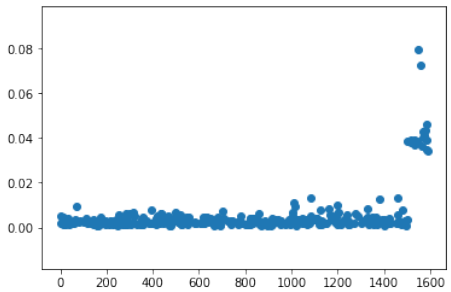
Pokud se podíváme na grafy ohodnocení testovacích dat po konci učení, můžeme vidět, jak se postupně vytvářely dva shluky oddělující běžná pozorování a odlehlá pozorování. Od čísla 6 dále se tyto shluky dále formovaly do kompaktnějšího uskupení.



Obrázek 3: 1 neuron

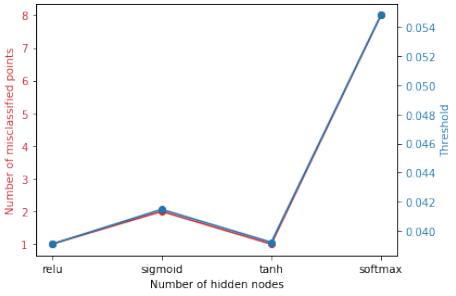


Obrázek 4: 10 neuronů



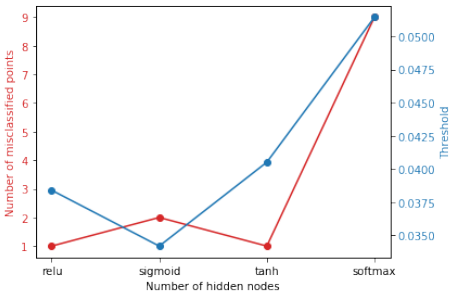
Obrázek 5: 30 neuronů

V druhé části jsem se zaměřil na výběr aktivační funkce a zajímalo mě, jestli bude mít její volba důsledek na výsledek. Následující grafy vznikly tak, že jsem vzal z předchozího měření první počet neuronů ve skryté vrstvě, který byl schopen s nulovou chybou klasifikovat trénovací dataset, tento počet byl 6, a pro tento počet jsem zkusil natrénovat autoenkodér s různými aktivačními funkcemi ve skryté vrstě.



Obrázek 6: Undercomplete autoenkodér

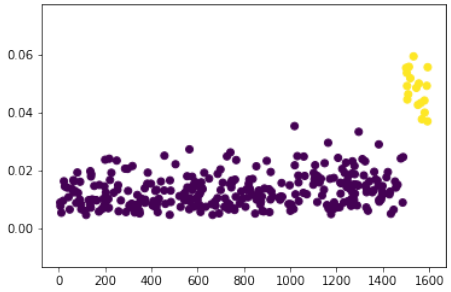
Z těchto grafů je vidět, že relu a tanh vrací prakticky stejné výsledky. Sigmoida



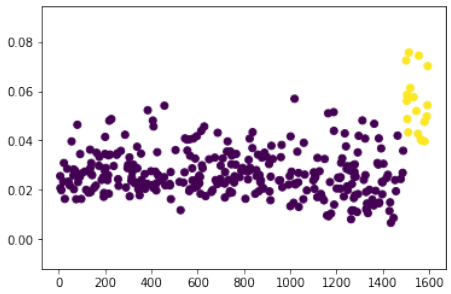
Obrázek 7: Sparse autoenkodér

vrací o trochu horší a použití softmaxu jako aktivační funkce je velice nevhodné.

Pokud do grafu vyneseme body a jejich ohodnocení pro jednotlivé aktivační funkce, můžeme vidět, jak pomocí relu a tanh je dataset uskupen do dvou skoro kompaktních bloků. Na druhé straně s použitím softmaxu se oba shluky hodně propojili.



Obrázek 8: Tanh



Obrázek 9: Softmax

Na úplný závěr jsem vyzkoušel další testovací datasety (Cardio a Satimage-2) a

vyzkoušel, jak bude autoenkodér schopný klasifikovat je. Výsledek je prakticky totožný s výsledekm výše. Po nalezení určité hranice byly oba autoenkodéry schopny klasifikovat datasety s nulovou chybou. Stejný výsledek mělo i vyzkoušení různých aktivačních funkcí.

3 Závěr

V rámci semestrální práce jsem si vyzkoušel tvorbu autoenkodéru a jeho využití ke klasifikaci odlehlých pozorování. Díky určitým znalostem a předpokladům se mi během práce podařilo rychle odhalit chyby během implementace a dovést práci do úspěšného konce.