FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská Univerzita v Plzni

Fakulta Aplikovaných Věd

Umělá inteligence a rozpoznávání

Integrace a testování klasifikačních metod pro deep learning

David Bohmann
A14B0229P

vedoucí práce
Ing. Lukáš Vařeka

10. května 2016

Obsah

1 Zadání

Přesné zadání projektu je převzato z dokumentu „Projekty ZSWI 2016“, jedná se o projekt číslo 23: Integrace a testování klasifikačních metod pro deep learning.

Na katedře informatiky provádíme výzkum v oblasti rozhraní mozek-počítač (BCI). K předvedení možností těchto systémů slouží aplikace Guess the number (Hádání čísel), `https://github.com/NEUROINFORMATICS-GROUP-FAV-KIV-ZCU/\discretionary{-}{}{}the_number`.

Měřená osoba si tajně vybere číslo 1 - 9 a soustředí se na něj. Následně se odebere do zvukotěsné komory, kde jí jsou snímány mozkové vlny. Zároveň se na monitoru pustí náhodná sekvence z těchto čísel. Program analyzuje mozkové odpovědi měřené osoby na jednotlivá čísla a s využitím klasifikačního algoritmu se pokouší uhodnout myšlené číslo, na které mozek reaguje jinak než na čísla ostatní.

Cílem práce je integrace a otestování moderních klasifikačních metod pro deep learning do této aplikace. Tyto algoritmy jsou už rozvíjeny v open-source Java knihovně deeplearning4j.org a není je tedy nutné samostatně implementovat. Cílem je však jejich napojení na existující rozhraní projektu Guess the number a hledání optimálních parametrů.

2 Analýza úlohy

2.1 Seznámení s vyvíjeným projektem

Projekt Guess the number je vyvíjen na KIV pod vedením Ing. Lukáše Vařeky. V projektu je již naimplementováno několik funkčních klasifikátorů s různou složitostí a úspěšností (KNN, MLP, SVN, atd.). Dále jsou naimplementované různé metody pro extrakci příznaků. Naším úkolem bylo naimplementovat dva nové klasifikátory s využitím knihovny deeplearning4j, konkrétně se jedná o klasifikátory Deep Belief Network (dále DBN) a Stacked Denoising Autoencoders (SDA).

Aplikace pracuje ve dvou režimech. Prvním je online režim, kdy se mozkové signály snímají člověku uzavřenému ve zvukotěsné komoře, kterému se promítá sekvence čísel 1 - 9. Aplikace může fungovat v offline režimu, kdy pracuje s již naměřenými daty.

Hlavní cíl této aplikace je na základě naměření a klasifikování dat uhodnout číslo, které si testovaná osoba myslí.

2.2 Úkoly

Náš tým měl v rámci projektu čtyři hlavní úkoly

- Konverze do Maven
- Implementace klasifikátorů
- Úprava GUI
- Testování úspěšnosti

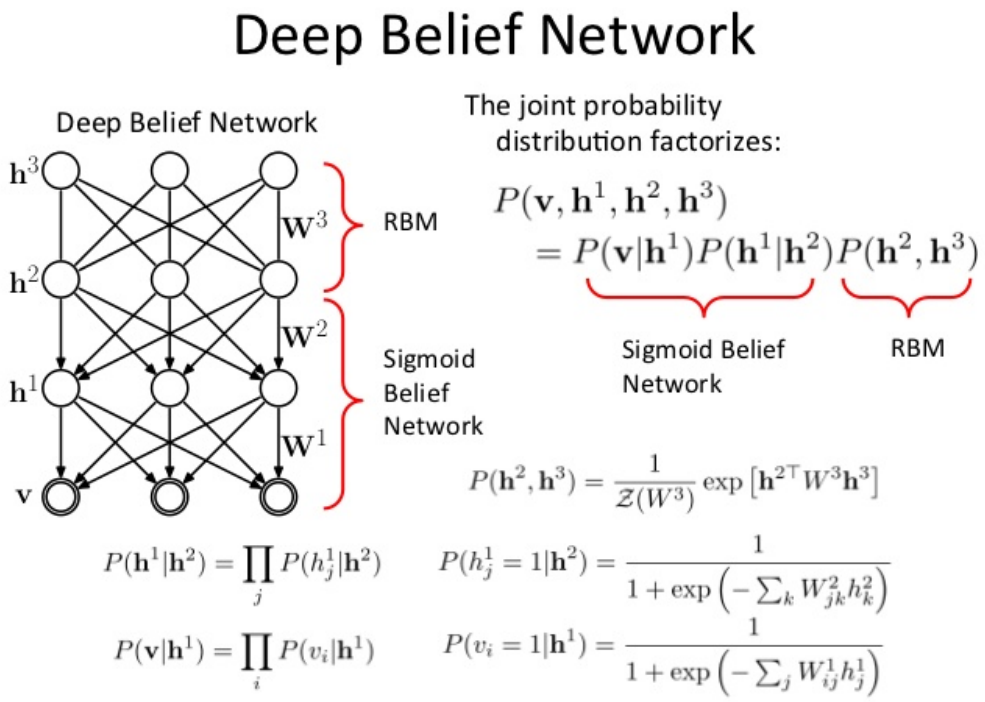
3 Klasifikátory

3.1 Deep Belief Network

Deep Belief Network (DBN) je typ hluboké neuronové sítě složené z několika vrstev se skrytými uzly, přičemž uzly jsou spojeny napříč vrstvami, ale nikoliv uvnitř jedné vrstvy (Obr. 1).

Námi implementovaný klasifikátor z deeplearning4j používá pouze jednu skrytou vrstvu „Restricted Bolzmann Machines“. Tato vrstva komunikuje s výstupní vrstvou „softmax“

Síť využívá trénování bez učitele, přičemž se může naučit odhadnout pravděpodobné vstupy. Po tomto kroku je možné provádět trénování s učitelem pro zlepšení úspěšnosti.



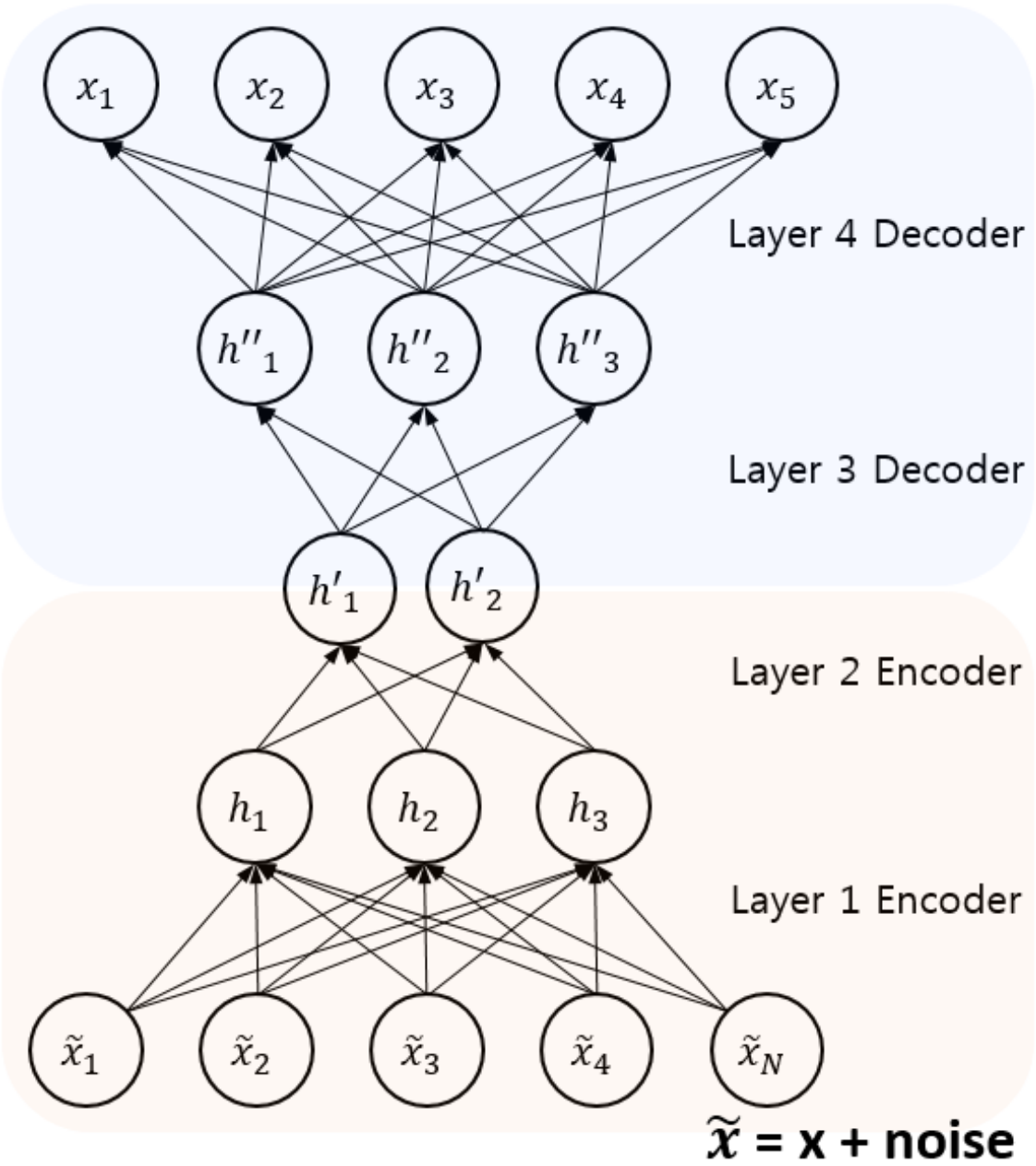
Obrázek 1: Schéma klasifikátoru Deep Belief Network

3.2 Stacked Denoising Autoencoders

Stacked Denoising Autoencoder je rozšíření původního Stacked Autencoders. Trénování tohoto klasifikátoru probíhá vrstvu po vrstvě tak, jak procházejí vstupní data (Obr. 2).

Cílem je, aby výstupní vrstva produkovala stejná data, která jsou na vstupu. Počet neuronů vstupní vrstvy je tedy roven počtu vstupů.

Trénování probíhá bez učitele. Data jsou náhodně promíchána a v procesu učení se snaží rekonstruovat původní konfiguraci - Denoising.



Obrázek 2: Schéma klasifikátoru Stacked Denoising Autoencoders

4 Popis implementace

4.1 Konverze do Maven

První krok našeho úkolu byla konverze celého projektu do Mavenu. Maven je je nástroj pro správu, řízení a automatizaci buildů aplikací. Běh a požadavky projektu výrazně zjednodušuje. Namísto importování externích .jar souborů využívá pouze konfiguračního souboru, ve kterém je popsáno, jaké knihovny jsou pro běh programu třeba. Projekt využívá přes 50 externích knihoven, což znamenalo velkou práci se sepsáním konfiguračního souboru `pom.xml`.

Některé knihovny bylo možno aktualizovat a zrychlit běh programu, některé ovšem musely zůstat v původní verzi z důvodu kompatibility.

4.2 Implementace klasifikátorů

Další, trochu složitější částí, bylo implementovat klasifikátory a zapojit je do stávajícího projektu.

V rámci celého projektu je již množství uložených dat, které jsou určeny pro trénování a testování vytvořených klasifikátorů. Velký problém jsme měli s napojením existujících dat do tříd využívajících Deep Learning. Bylo třeba z klasických dvourozměrných polí vytvořit vektory NDAarray z knihovny Nd4j pro vstupní a výstupní data. Tyto vektory byly následně využity pro vytvoření DataSetu, který je použit pro trénování daného klasifikátoru.

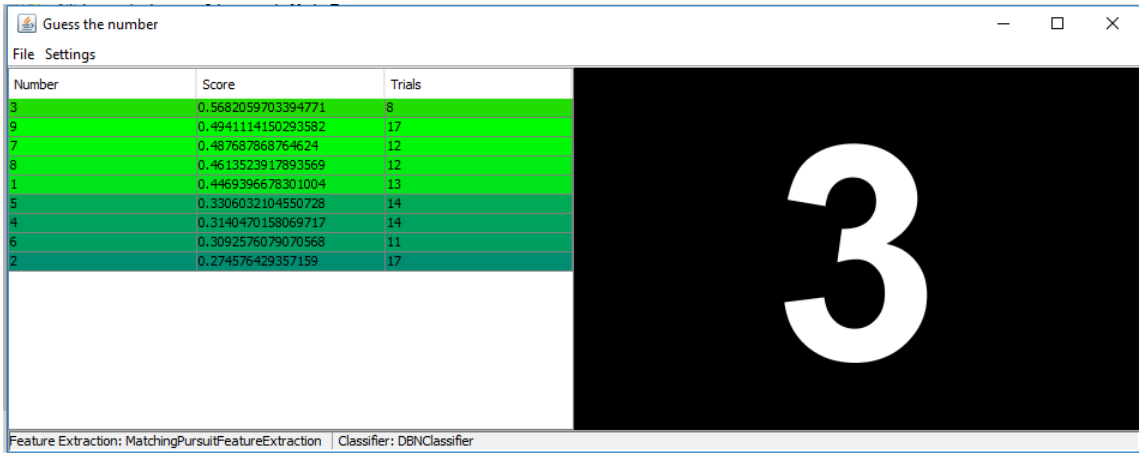
Samotné vytváření neuronové sítě nebylo hlavní náplní naší práce, tato část kódu je vyvíjena na stránkách `http://deeplearning4j.org/`. Oba klasifikátory fungují na principu MultilLayerNetwork. V rámci testování jsme v této části měnili parametry tak, aby síť dosahovala v našem projektu co nejlepší úspěšnosti.

4.3 Úprava GUI

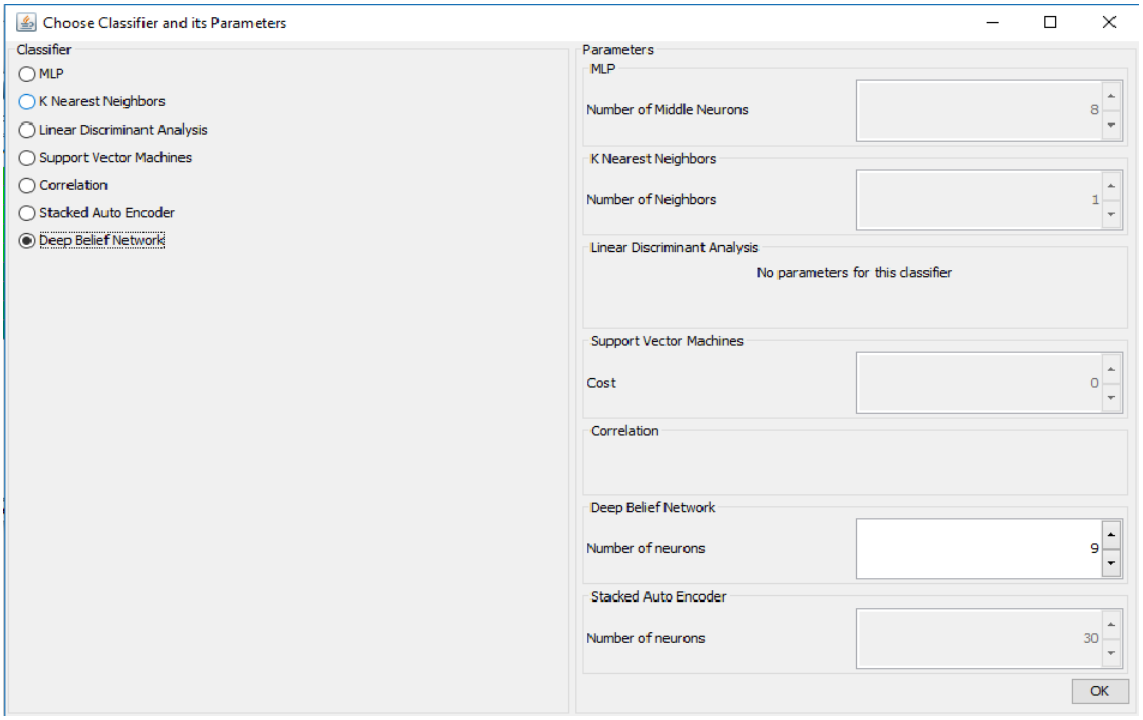
Projekt má základní uživatelské rozhraní, které na hlavní obrazovce zobrazuje průběh klasifikace (Obr. 3). Je možné navolit si parametry klasifikace a dle nich vytvořit nový klasifikátor. Dále je možné vybrat pro klasifikaci již natrénované klasifikátory. Tyto klasifikátory jsou uloženy v souborech `<nazev>.classifier` a načítají se pomocí `<nazev>.txt` souboru, ve kterém jsou uložené parametry.

Námi implementované klasifikátory si vytváří pomocný soubor s konfigurací `<nazev>.bin`. Bylo třeba vyřešit to, aby se při ukládání a načítání zvolil správný soubor s konfigurací.

V rámci GUI jsme museli změnit část, kde se vybírá klasifikátor a nastavují se jeho parametry (Obr. 4).



Obrázek 3: GUI - Klasifikace



Obrázek 4: GUI - Výběr klasifikátoru

4.4 Testování úspěšnosti

V projektu je již napsáno několik tříd, které se využívají pro otestování úspěšnosti klasifikátorů. Je nutno pouze nastavit, jak chceme extrahovat příznaky a jak chceme klasifikovat. Výsledkem testování je výpis statisticých údajů, které zahrnují, kolik případů bylo správně a chybně ohodnoceno a celkovou procentuální úspěšnost.

V rámci tříd klasifikátorů jsme nastavovali počet vrstev, počet neuronů, kontrastivní divergenci, rychlost učení, a mnoho dalších parametrů. Největší vliv na úspěšnost má parametr počet neuronů v síti.

Testovací třídy ohodnotí klasifikátory dle dvou kritérií:

- Rozpoznání příznakové vlny P300 (vlna by se měla objevit, pokud se zobrazí číslo, které si daný člověk myslí). Tato klasifikace probíhá na principu ANO/NE, čili úspěšný klasifikátor se musí pohybovat nad hranicí úspěšnosti 50%.

- Uhodnutí myšleného čísla (číslo, u kterého člověk reagoval mozkovou aktivitou - příznakovou vlnou P300). Čísla jsou od 1 do 9, což znamená, že úspěšný klasifikátor se musí pohybovat nad hranicí úspěšnosti 11,1%.

Klasifikátor DBN:

- Rozpoznání příznakové vlny P300: 75%

- Uhodnutí myšleného čísla : 64%

Klasifikátor SDA:

- Rozpoznání příznakové vlny P300: 86%

- Uhodnutí myšleného čísla : 75%

5 Uživatelská dokumentace

5.1 HW a SW požadavky

Celý projekt musí fungovat na počítači, který má následující minimální konfiguraci: Intel Core i7 2.70 GHz, RAM 4 GB, 64-bitový systém. Program musí fungovat na Windows 7 a vyšší s nainstalovanou Javou JRE 1.8 a Apache Maven.

5.2 Návod k použití

Projekt je vyvíjen na GitHubu, pro spuštění je třeba ho stáhnout z adresy `https://github.com/Pumprdlici/hadej_cislo`. Pro spuštění v Eclipse je třeba importovat jako Maven projekt.

Pro projekt se jako defaultní kompilátor nastaví Java 1.5, což je třeba změnit v *Project - Properties - Java Compiler* na Java 1.8. Poté již lze bez problémů spustit třídu Main ve složce *src/icp*.

Po spuštění programu se objeví vyskakovací okno nabízející výběr již natrénovaného klasifikátoru ze souboru `<nazev>.txt`. Ve složce *data/test_classifiers_and_settings* jsou uloženy natrénované klasifikátory. Ve složce *data/new_models* jsou uloženy natrénované klasifikátory s nejlepší úspěšností. Uživatel si může vybrat již natrénovaný klasifikátor, nebo v menu *Settings - Feature Extraction and Classifier* vytvořit klasifikátor dle vlastních požadavků.

Po zvolení klasifikátoru uživatel načítá data, která chce klasifikovat. Program funguje v „Online módu“, při kterém je napojen přímo na zvukotěsnou komoru, ve které se nachází testovaná osoba, nebo v „Offline módu“, kdy jsou ke klasifikaci použita již uložená data ve formátu `<data>.eeg`. Pro načtení uložených dat je třeba zvolit *File - Offline data*. Uložená data jsou ve složce *data/numbers/....*

Program dokáže graficky zobrazit, jak vypadají vstupní data - mozkové signály. Je třeba zvolit *File - Show charts*.

6 Závěr

Věřím, že v rámci semestrální práce se nám podařilo úspěšně splnit všechny body zadání. Myslím si, že úspěšnost klasifikátoru by mohla být mnohem lepší, bohužel optimalizace nebyla součástí našeho zadání a myslím, že by měla mnohem větší časovou náročnost.

Závěrem bych rád řekl, že toto byl první větší projekt, na kterém jsem měl příležitost se podílet. Odnáším si z toho mnoho pozitivních, ale i nějaké negativní zkušenosti.

Jsem velmi rád, že jsem si mohl vyzkoušet, jak probíhá práce na týmovém projektu, co všechno je třeba dělat kromě psaní kódu. Věřím, že se mi velmi bude hodit zkušenost s nástrojem Maven. Velice přínosné je seznámení s hlubokými neuronovými sítěmi Deep Learning, přestože jsou stále ještě v začátcích a jsou velmi rychle vyvíjeny, myslím, že v budoucnu se budou využívat stále častěji.

Naopak nemile mne překvapilo, kolik času jsme museli strávit seznamováním s původním projektem, který jsme měli rozšířit. V projektu byl opravdu „nepořádek“ a bez důkladné uživatelské dokumentace (kterou jsme k dispozici neměli) bylo zezačátku velmi těžké se v něm vyznat.