Mateusz Woźniak 22.06.2023

Sprawozdanie lista 5 – Sztuczna inteligencja i inżynieria wiedzy

1. Dane

Dane użyte w zadaniach to zbiór nr 1 *Jester* ze strony https://eigentaste.berkeley.edu/dataset/

Zawiera on 100 plików o rozszerzeniu .html, które zawierają żarty.

Ponadto w pliku *jester-data-1.xls* występują ich oceny dokonane przez użytkowników, gdzie:

- każdy użytkownik ocenił przynajmniej 36 żartów
- oceny są w skali -10.0 – 10; brak oceny jest oznaczany wartością 99
- każdy numer kolumny odpowiada żartowi o tym samym numerze
- dane pochodzą od 24983 użytkowników pomiędzy kwietniem 1999 a majem 2003
- łącznie zawiera 4,1 miliona ocen

2. Rozwiązania zadań

- Przygotuj zbior uczący i walidacyjny, wykorzystując dołączony do listy kod procedury ekstrakcji cech.

```python
if __name__=="__main__":
    ratings = pd.read_excel("jester-data-1.xls", header=None)
    ratings = ratings.iloc[:, 1:].replace(99, float('nan'))

    avgRatings = ratings.mean()

    jokes = []

    jokesCount = len(os.listdir("jokes"))

    for i in range(1, jokesCount):
        with open(f"jokes/init{i}.html") as joke:
            joke_html = joke.read()
            soup = BeautifulSoup(joke_html, 'html.parser')
            joke_text = soup.find('font', size='+1').text.strip()
            jokes.append(joke_text)

    model = SentenceTransformer('bert-base-cased')
    embeddings = model.encode(jokes)

    X_train, X_validate, y_train, y_validate = train_test_split(
        embeddings,
        avgRatings,
        test_size=0.2,
        random_state=RANDOM_STATE
    )
```

Po odczytaniu danych, zastąpiłem wartości 99 nieokreślonymi, aby sztucznie nie zawyżały średnich. Następnie obliczono średnie i zapisano w zmiennej avgRatings. Należało dokonać ekstrakcji żartów z plików html i w tym celu skorzystano z BeautifulSoup. Po skorzystaniu z podanego w zadaniu kodu procedury ekstrakcji cech, podzielono dane na zbiory: uczący i walidacyjny.

- Przetestuj działanie podstawowego modelu MLP o domyślnej konfiguracji hiperparametrow, ucząc go na danych ze zbioru Jester. Prześledź zachowanie modelu w czasie, wizualizując wartość funkcji kosztu w funkcji liczby epok, zwracając uwagę na wartości dla zbioru uczącego i zbioru walidacyjnego.

Kod wykorzystany do rozwiązania tego, oraz następnych zadań składał się z:

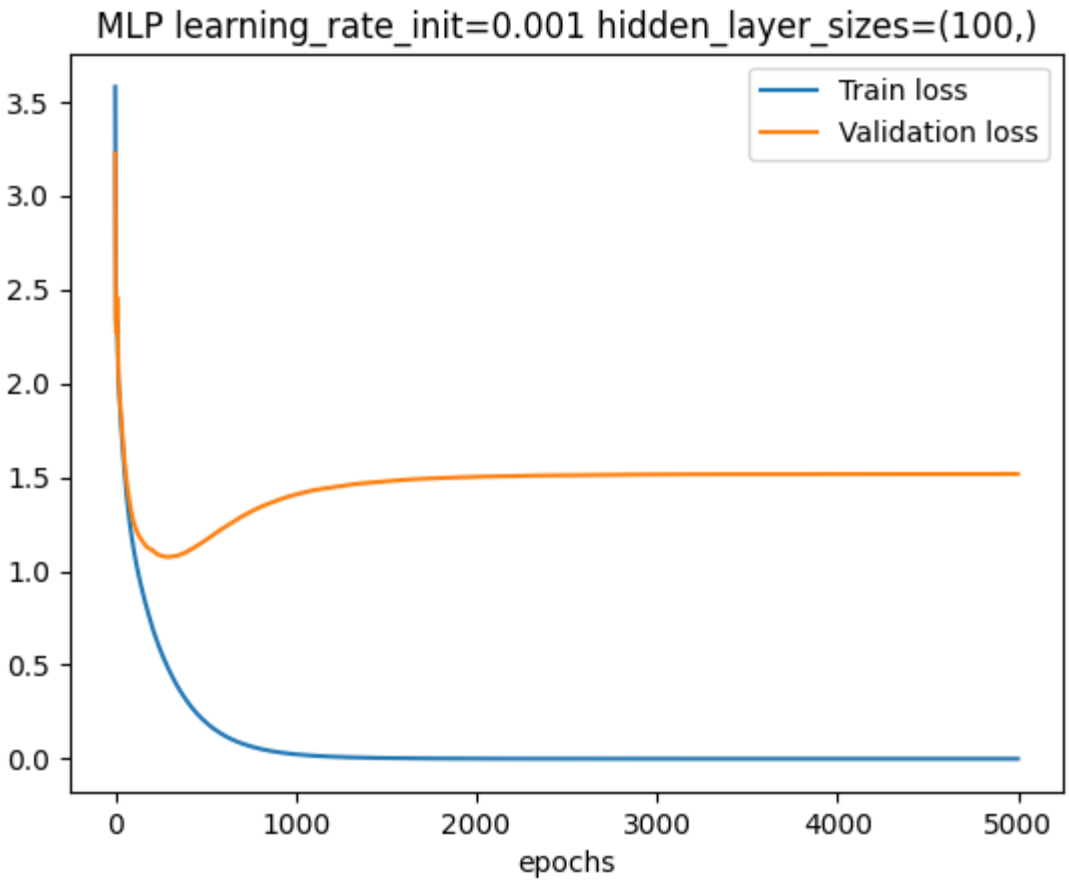
```python
def createPlots(xt, xv, yt, yv):
    learning_rates = [0.001, 0.005, 0.0005]
    hidden_sizes_list = [(100,), (200,), (100, 100,), (100, 100, 100)]

    for lr in learning_rates:
        tl, vl = testMLP(xt, xv, yt, yv, learning_rate=lr, hidden_sizes=hidden_sizes_list[0])
        plt.plot(tl, label='Train loss')
        plt.plot(vl, label='Validation loss')
        plt.xlabel('epochs')
        plt.title(f"MLP learning_rate_init={lr} hidden_layer_sizes={hidden_sizes_list[0]}")
        plt.legend()
        plt.show()

    for hs in hidden_sizes_list:
        tl, vl = testMLP(xt, xv, yt, yv, learning_rate=learning_rates[0], hidden_sizes=hs)
        plt.plot(tl, label='Train loss')
        plt.plot(vl, label='Validation loss')
        plt.xlabel('epochs')
        plt.title(f"MLP learning_rate_init={learning_rates[0]} hidden_layer_sizes={hs}")
        plt.legend()
        plt.show()
```

Rysunek 1 Metoda tworząca wykresy

```python
def testMLP(xt, xv, yt, yv, epochs=5000, learning_rate=0.001, hidden_sizes=(100,)):
    mlp = MLPRegressor(
        solver='sgd',
        alpha=0.0,
        learning_rate_init=learning_rate,
        random_state=RANDOM_STATE,
        hidden_layer_sizes=hidden_sizes
    )

    train_loss = []
    validation_loss = []

    for _ in range(epochs):
        mlp.partial_fit(xt, yt)

        pred_y_train = mlp.predict(xt)
        train_loss.append(mean_squared_error(y_train, pred_y_train))

        pred_y_validation = mlp.predict(X_validate)
        validation_loss.append(mean_squared_error(yv, pred_y_validation))

    return train_loss, validation_loss
```

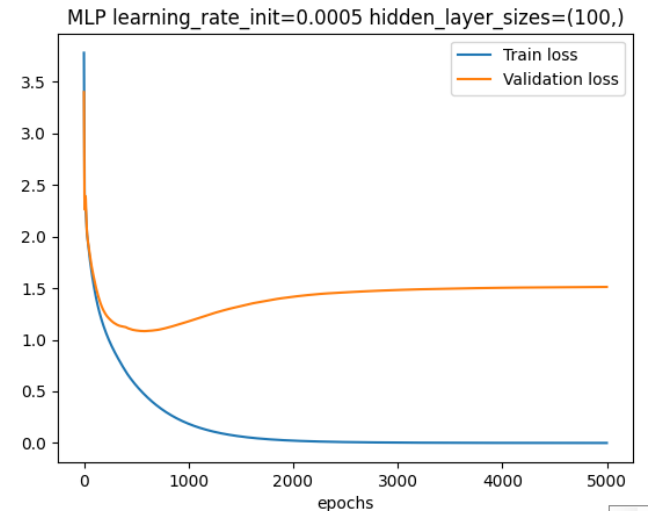
Rysunek 2 Funkcja testująca modele MLP

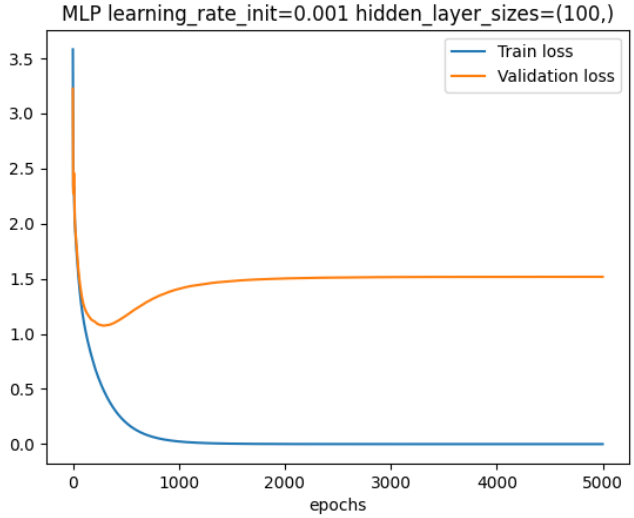
Po uruchomieniu ww. kodu dla domyślnych parametrów (określone w tytule tego wykresu) otrzymano poniższy wykres z wyróżnieniem funkcji kosztu w zbiorze uczącym i walidacyjnym.

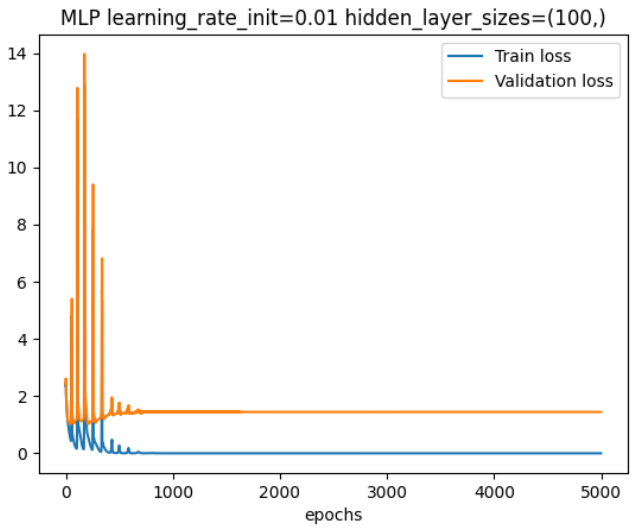


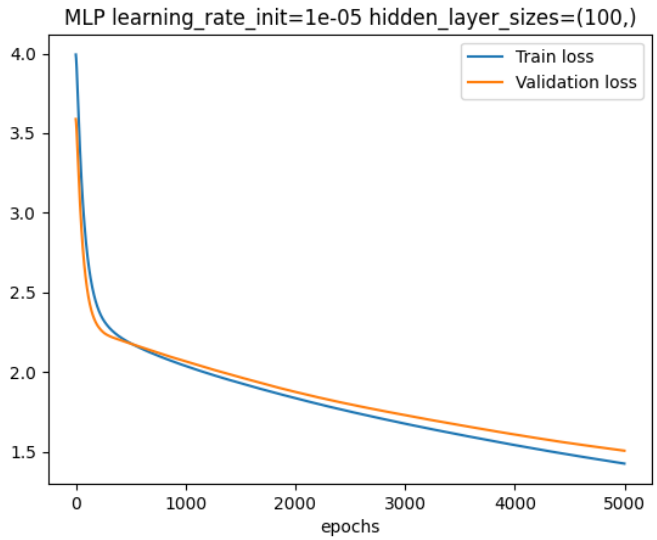
Oba zbiory mają podobną tendencje do około 200 epok. Następnie funkcja kosztu w zbiorze walidacyjnym lekko „odbija się”, zaś funkcja kosztu zbioru uczącego zmierza do zera i na poziomie ok. 800 epok osiąga tę wartość.

- Zbadaj wpływ tempa uczenia (learning rate) na osiągane wyniki: powtorz uczenie dla 3 rożnych wartości parametru. Dobierz odpowiednią długość procesu uczenia (liczbę epok) jeśli to konieczne. Przedstaw wyniki na wykresach jak w zadaniu poprzednim. Co dzieje się, gdy tempo uczenia jest zbyt niskie? Co, gdy zbyt wysokie?



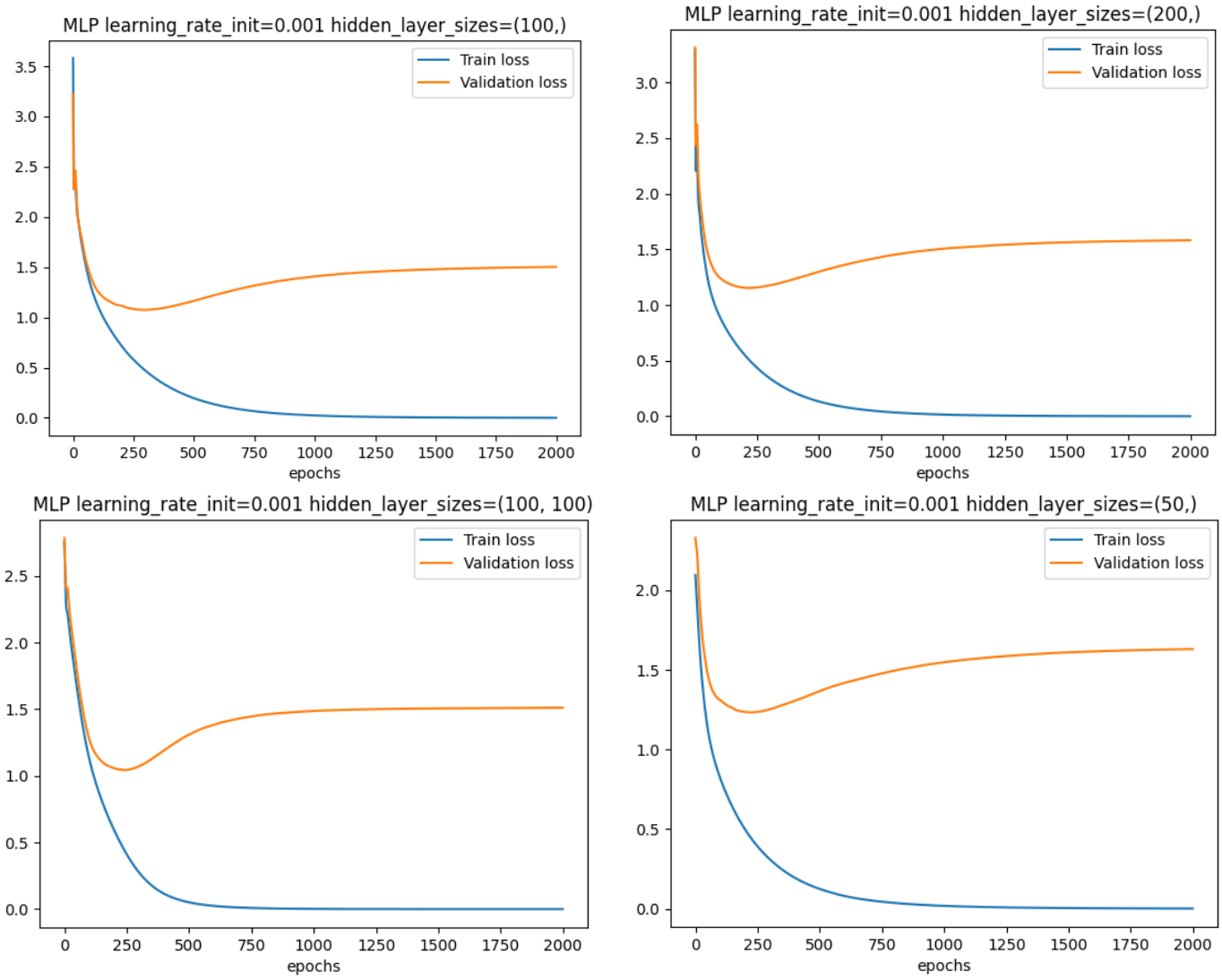






Dla zbyt wysokiego tempa uczenia obie wartości momentami są ogromne. Szum jest coraz mniejszy wraz z obniżaniem tempa uczenia. Gdy jest ono bardzo niskie, model ma skłonność do zjawiska znanego jako *underfitting* ze względu na wysoki *Training Loss*. W pozostałych przypadkach, czyli tych pomiędzy wskazanymi skrajnościami, model dokonuje *overfittingu – Validation Loss* jest znacznie wyższy od *Training Loss'a* prawie na całym przedziale epok.

- Zbadaj wpływ rozmiaru modelu MLP na jakość działania: wykonaj co najmniej 3 eksperymenty dla modeli rożniących się liczbą neuronow. Kiedy model przestaje dobrze dopasowywać się do danych? Kiedy zaczyna zanadto dopasowywać się do zbioru uczącego?



Wraz ze wzrostem liczby neuronów oraz warstw neuronów, model prezentuje podobne wyniki, lecz potrzebuje do tego mniejszej liczby epok. W każdym z tych przypadków *Overfitting* występuje od momentu pomiędzy 100 a 400 epok i wydaje się być o wręcz identycznej skali, jednak w przypadku 50 neuronów jest on nieznacznie większy.

- Wybierz najlepszy uzyskany w drodze powyższych eksperymentow model i przetestuj go w praktyce: znajdź (lub napisz własny) tekst o charakterze dowcipu, przetworz go na wektor za pomocą używanej w zadaniach metody ekstrakcji cech, a następnie odpytaj model neuronowy. Czy predykcja zgadza się z Twoim oczekiwaniem?

  Do wykonania tego zadania utworzyłem dodatkową metodę, która losowo wybiera żart, prezentuje go oraz dokonuje predykcji:

```
def testMyJoke(X_train, X_validate, y_train, y_validate, jokes, joke=None):
    mlp = MLPRegressor(
        solver='sgd',
        alpha=0.0,
        learning_rate_init=0.00001,
        random_state=RANDOM_STATE
    )
    mlp.fit(X_train, y_train)
    mlp.score(X_validate, y_validate)

    if(joke is None):
        index = int(random.random() * len(jokes))
        myJoke = str(jokes[index])
    else:
        myJoke = joke
    print(myJoke)
    model = SentenceTransformer('bert-base-cased')
    embeddings = model.encode([myJoke])
    embeddings = np.reshape(embeddings, (1,-1))
    prediction = mlp.predict(embeddings)
    print(f"Predykcja -> {prediction}")
```

Żart	Predykcja
At a recent Sacramento PC Users Group meeting, a company was demonstrating its latest speech-recognition software. A representative from the company was just about ready to start the demonstration and asked everyone in the room to quiet down. Just then someone in the back of the room yelled, "Format C: Return." Someone else chimed in: "Yes, Return"	0.8073065
Co jest męczące, krótkie, choć zabiera cały piątek? Eegzamin z Rozproszonych Systemów Informatycznych	0.42588663
During a recent publicity outing, Hillary sneaked off to visit a fortune teller of some local repute. In a dark and hazy room, peering into a crystal ball, the mystic delivered grave news. "There's no easy way to say this, so I'll just be blunt: Prepare yourself to be a widow. Your husband will die a violent and horrible death this year." Visibly shaken, Hillary stared at the woman's lined face, then at the single flickering candle, then down at her hands. She took a few deep breaths to compose herself. She simply had to know. She met the fortune teller's gaze, steadied her voice, and asked her question. "Will I be acquitted?"	0.83252406

Z racji na to, że zbiór uczył się na ocenach kilkudziesięciu tysięcy użytkowników, zrozumiałe jest, że mój żart o RSI mógłby do nich nie trafić. Ten żart głównie śmieszyłby studentów Informatyki Stosowanej na Politechnice Wrocławskiej. Pozostałe dwa żarty nie były bardzo złe a ich predykcja była w obrębie moich oczekiwań.