

Laboratorium: Klasyfikacja i regresja w Keras

May 26, 2022

1 Zakres ćwiczeń

- Przypomnienie zasad budowy prostych sieci neuronowych dla klasyfikacji oraz regresji.
- Budowanie sieci przy pomocy API Sequential Keras.
- Zbieranie danych procesu uczenia i jego wizualizacja przy pomocy TensorBoard.
- Korzystanie z mechanizmu *early stopping* do zatrzymania procesu uczenia w odpowiednim czasie.

2 Zadania

2.1 Klasyfikacja obrazów

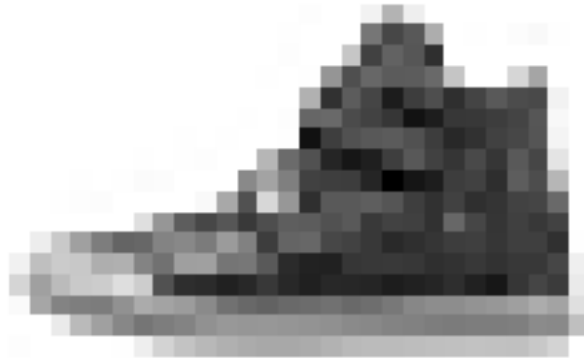
Pobierz zbiór danych *Fashion MNIST*.

```
import tensorflow as tf
fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
assert X_train.shape == (60000, 28, 28)
assert X_test.shape == (10000, 28, 28)
assert y_train.shape == (60000,)
assert y_test.shape == (10000,)
```

Przeskaluj wartości z zakresu 0–255 do zakresu 0–1.

Wyświetl przykładowy rysunek używany do klasyfikacji:

```
import matplotlib.pyplot as plt
plt.imshow(X_train[142], cmap="binary")
plt.axis('off')
plt.show()
```



Utwórz listę nazw kategorii zgodnie ze specyfikacją zbioru danych:

```
class_names = ["koszulka", "spodnie", "pulower", "sukienka", "kurtka",  
               "sandał", "koszula", "but", "torba", "kozak"]  
class_names[y_train[142]]
```

'but'

Stwórz model sekwencyjny zawierający warstwy gęste:

- warstwę spłaszczającą dane, tj. przekształcającą z postaci (28,28) do postaci (784),
- warstwę ukrytą zawierającą 300 neuronów,
- warstwę ukrytą zawierającą 100 neuronów,
- warstwę wyjściową odpowiednią dla problemu klasyfikacji przy 10 klasach.

Wyświetl podsumowanie i graficzną reprezentację struktury sieci.

```
model.summary()  
tf.keras.utils.plot_model(model, "fashion_mnist.png", show_shapes=True)
```

Skompiluj model, podając rzadką entropię krzyżową jako funkcję straty, SGD jako optymalizator i dokładność jako metrykę.

Przygotuj callback Tensorboard do zbierania historii uczenia w katalogu image_logs, będącym podkatalogim bieżącego katalogu. Można do tego użyć takiej funkcji jak poniżej, która przy każdym wywołaniu będzie zwracała nowy katalog o nazwie opartej o aktualny czas. Alternatywnie możesz nazywać katalogi np. oznaczając w nazwie informacje o strukturze sieci. Pamiętaj, że TensorBoard zawsze szuka w podanym katalogu podkatalogów odpowiadających kolejnym procesom uczenia sieci.

```

import os

root_logdir = os.path.join(os.curdir, "image_logs")

def get_run_logdir():
    import time
    run_id = time.strftime("run_%Y_%m_%d-%H_%M_%S")
    return os.path.join(root_logdir, run_id)

run_logdir = get_run_logdir()

tensorboard_cb = tf.keras.callbacks.TensorBoard(run_logdir)

```

Przyucz model przez 20 epok. Wykorzystaj 10% zbioru uczącego jako zbioru walidacyjnego – zwróć uwagę, że nie został on wcześniej wydzielony ręcznie. Pamiętaj o dołączeniu callbacku TensorBoard.

Wykonaj kilka przykładowych predykcji dla losowych elementów zbioru testowego. Czy podpisy odpowiadają zawartości obrazków?

```

image_index = np.random.randint(len(X_test))
image = np.array([X_test[image_index]])
confidences = model.predict(image)
confidence = np.max(confidences[0])
prediction = np.argmax(confidences[0])
print("Prediction:", class_names[prediction])
print("Confidence:", confidence)
print("Truth:", class_names[y_test[image_index]])
plt.imshow(image[0], cmap="binary")
plt.axis('off')
plt.show()

```

```

Prediction: kurtka
Confidence: 0.4835742
Truth: kurtka

```



Przeanalizuj proces uczenia uruchamiając TensorBoard. Możesz to zrobić albo uruchamiając jego proces w Terminalu

```
$ tensorboard --logdir=./image_logs --port=6006
```

i otwierając jego interfejs przy pomocy odpowiedniego adresu w przeglądarce, lub korzystając z rozszerzenia Jupytera, najpierw ładując samo rozszerzenie:

```
%load_ext tensorboard
```

i następnie uruchamiając sam TensorBoard:

```
%tensorboard --logdir ./image_logs
```

Zapisz model w pliku `fashion_clf.h5`:

```
model.save('fashion_clf.h5')
```

2.2 Regresja

Pobierz zbiór danych California Housing z pakietu scikit-learn:

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

housing = fetch_california_housing()
```

Podziel zbiór na uczący, walidacyjny i testowy. Zauważ, że tym razem wydzielamy zbiór walidacyjny *explicite* i nie będziemy korzystali z argumentu `validation_split` metody `fit`.

```
X_train_full, X_test, y_train_full, y_test = train_test_split(housing.data,
    ↪housing.target, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full,
    ↪y_train_full, random_state=42)
```

Przeskaluj wszystkie zbiory cech, kalibrując funkcję normalizacyjną do zbioru uczącego:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

Utwórz model, zawierający jedną warstwę ukrytą z 30 neuronami, odpowiedni do regresji jednego parametru wyjściowego obecnego w zbiorze danych. Skompiluj go używając błędu średniokwadratowego jako funkcji straty i SGD jako optymalizatora.

Przygotuj callback *early stopping* o cierpliwości równej 5 epok, minimalnej wartości poprawy wynoszącej 0.01 i włączając wyświetlanie komunikatów o przerwaniu uczenia na ekranie:

```
es = tf.keras.callbacks.EarlyStopping(patience=5,
    min_delta=0.01,
    verbose=1)
```

Podobnie jak w poprzednim ćwiczeniu, przygotuj callback Tensorboard, tak aby zbierał logi do katalogu `housing_logs`.

Przeprowadź uczenie modelu korzystając z obu callbacków (*early stopping* i TensorBoard). Jaką liczbę epok uczenia należy podać w tym przypadku?

Zapisz model w pliku `reg_housing_1.h5`.

Utwórz jeszcze co najmniej dwa modele o innej strukturze – poeksperymentuj z liczbą warstw oraz liczbą jednostek na jednej warstwie. Zapisz je w plikach `reg_housing_2.h5`, `reg_housing_3.h5`, itd. Zapisuj logi TensorBoard w kolejnych podkatalogach katalogu `housing_logs`.

Uruchom TensorBoard i porównaj przebieg procesu uczenia w zależności od dobranych parametrów.