Laboratorium: Klasyfikacja i regresja w Keras

May 9, 2025

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*.

```
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.figure(figsize = (2,2))
plt.imshow(X_train[42], 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", "półbut", "torba", "but"]
class_names[y_train[42]]
```

```
'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 struktury i liczby parametrów sieci.

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 300)	235,500
dense_1 (Dense)	(None, 100)	30,100
dense_2 (Dense)	(None, 10)	1,010

```
Total params: 266,610 (1.02 MB)
```

```
Trainable params: 266,610 (1.02 MB)
```

```
Non-trainable params: 0 (0.00 B)
```

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

Przygotuj callback Tensorboard do zbierania historii uczenia w katalogu `image_logs`, który będzie podkatalogiem bieżącego katalogu.

Ponieważ nierzadko przeprowadzamy więcej niż jeden proces trenowania sieci, częstą praktyką jest automatyczne tworzenie katalogów dla kolejnych przebiegów, na przykład z nazwami opartymi o aktualny *timestamp*. Możesz użyć do tego na przykład takiej funkcji:

```
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()
```

Alternatywnie możesz nadawać katalogom nazwy związane np. z parametrami danego eksperymentu. Pamiętaj, że TensorBoard po uruchomieniu zawsze szuka w podanym katalogu podkatalogów, które odpowiadają kolejnym podejściom do trenowania sieci, i z ich zawartości generuje wizualizowane treści.

Utwórz callback TensorBoard i pamiętaj o dołączeniu go później w liście przekazywanej jako argument `callbacks` metody `fit()`.

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

Przyucz model przez 20 epok. Wykorzystaj 10% zbioru uczącego jako zbiór walidacyjny – zwróć uwagę że, w odróżnieniu od przykładu z wykładu czy podręcznika, 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.figure(figsize = (2,2))
plt.imshow(image[0], cmap="binary")
plt.axis('off')
plt.show()
```

```
1/1             0s 19ms/step
Prediction: półbut
Confidence: 0.5991093
Truth: półbut
```

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
```

Jeżeli jednak robisz to w notebooku, pamiętaj o zakomentowaniu tych linijek przy eksporcie do skryptu uruchamianego w trybie wsadowym.

Korzystając z metody `model.save()` zapisz model w pliku `fashion_clf.keras`.

6 punktów

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

housing = fetch_california_housing()
```

Podziel zbiór na uczący, walidacyjny i testowy. Tym razem ręcznie wydziel zbiór walidacyjny (`train_test_split`) i nie korzystaj z argumentu `validation_split` metody `fit`.

Utwórz model, zawierający:

- warstwę normalizującą wartości cech,
- 3 warstwy po 50 neuronów,
- warstwę wyjściową odpowiednią do regresji pojedynczej wartości.

Skompiluj go używając błędu średniokwadratwego jako funkcji straty i optymalizatora Adam, wyliczając dodatkowo RMSE jako metrykę. Pamiętaj o skalibrowaniu warstwy normalizacyjnej.

W praktyce często stosujemy mechanizm zwany *early stopping*, który zatrzymuje proces uczenia jeżeli przez określoną liczbę epok nie nastąpi wskazana poprawa wartości funkcji straty. 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.keras`.

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.keras`, `reg_housing_3.keras`, itd. Zapisuj logi TensorBoard w kolejnych podkatalogach katalogu `housing_logs`.

6 punktów (po 2 za każdy z modeli)

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

3 Wyślij rozwiązanie

Umieść skrypt realizujący powyższe ćwiczenia w pliku `lab09/lab09.py`.

Pamiętaj o przetestowaniu skryptu i upewnieniu się, że jest w stanie poprawnie pracować w trybie wsadowym i nie zawiera żadnych poleceń działających tylko w środowisku IPython/Jupyter.