Sprawozdanie Lab 10

Piotr Gruda

Zadanie 1.

Metoda prostokątów.

- Kod:

```
def integrate(function, a, b, i):
    dx = (b - a) / i
    integr = 0
    for x in range(i):
        x = x * dx + a
        integr += dx * eval(function)
    return integr

functio1n = input("Funkcja: ")
a = float(input("Początek przedziału: "))
b = float(input("Koniec przedziału: "))
i = int(input("Liczba podprzedziałów"))
print("Całka z funkcji {funkcjon} po przedziale od {a} do {b} =
{integrate}".format(funkcjon = function, a = a, b = b, integrate =
integrate(function, a, b, i)))
```

- Wynik:

```
Funkcja: x**2
Początek przedziału: 0
Koniec przedziału: 1
Liczba podprzedziałów10
Całka z funkcji x**2 po przedziale od 0.0 do 1.0 = 0.2850000000000001
```

Metoda trapezów.

- Kod:

```
def integrate(function, a, b, i):
    dx = (b - a) / i
    integr = 0
    for x in range(i):
        x = x * dx + a
        fx1 = eval(function)
        x += dx
        fx2 = eval(function)
        integr += 0.5 * dx * (fx1 + fx2)
    return integr

function = input("Funkcja: ")
a = float(input("Początek przedziału: "))
b = float(input("Koniec przedziału: "))
i = int(input("Liczba podprzedziałów: "))
print("Całka z funkcji {funkcjon} po przedziale od {a} do {b} =
{integrate}".format(funkcjon = function, a = a, b = b, integrate =
integrate(function, a, b, i)))
```

- Wynik:

```
Funkcja: x**2
Początek przedziału: 0
Koniec przedziału: 1
Liczba podprzedziałów: 10
Całka z funkcji x**2 po przedziale od 0.0 do 1.0 = 0.3350000000000001
```

Metoda Simpsona.

- Kod:

```
def simpson_integration_simple(my_func, a, b, n):
    delta_x = (b-a)/n
    total = 0
    for i in range(0, n, 2):
        x = a + delta_x * 2 * i
        total += delta_x * (my_func(x) + 4 * my_func(x + delta_x) +
my_func(x + 2 * delta_x)) / 3
    return total

integral = simpson_integration_simple(lambda x: x**2, 0.0, 1.0, 100)
integral
```

- Wynik:

```
1.3133333333333335
```

Zadanie 2.

Interpolacja a)

- Kod:

```
import tensorflow as tf
import numpy as np
from tensorflow import keras

def funckja(x):
  y = (x - 2) * (x + 1) * (x - 4)
  return y

xs = np.empty([6], dtype=float)
ys = np.empty([6], dtype=float)
for i in range(-1,5):
  xs[i + 1] = i
  ys[i + 1] = funckja(i)

model = tf.keras.Sequential([keras.layers.Dense(units=1,
input_shape=[1])])

model.compile(optimizer='sgd', loss='mean_squared_error')

model.fit(xs, ys, epochs=500)

print(model.predict([1]))
# print(xs, ys)
```

- Wynik:

```
1/1 [==============================] - 0s 4ms/step - loss: 12.3558
Epoch 492/500
1/1 [==============================] - 0s 4ms/step - loss: 12.3558
Epoch 493/500
1/1 [==============================] - 0s 4ms/step - loss: 12.3558
Epoch 494/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3558
Epoch 495/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3558
Epoch 496/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3558
Epoch 497/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3557
Epoch 498/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3557
Epoch 499/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3557
Epoch 500/500
1/1 [==============================] - 0s 3ms/step - loss: 12.3557
[[2.254715]]
```

Zadanie 3.

- Kod:

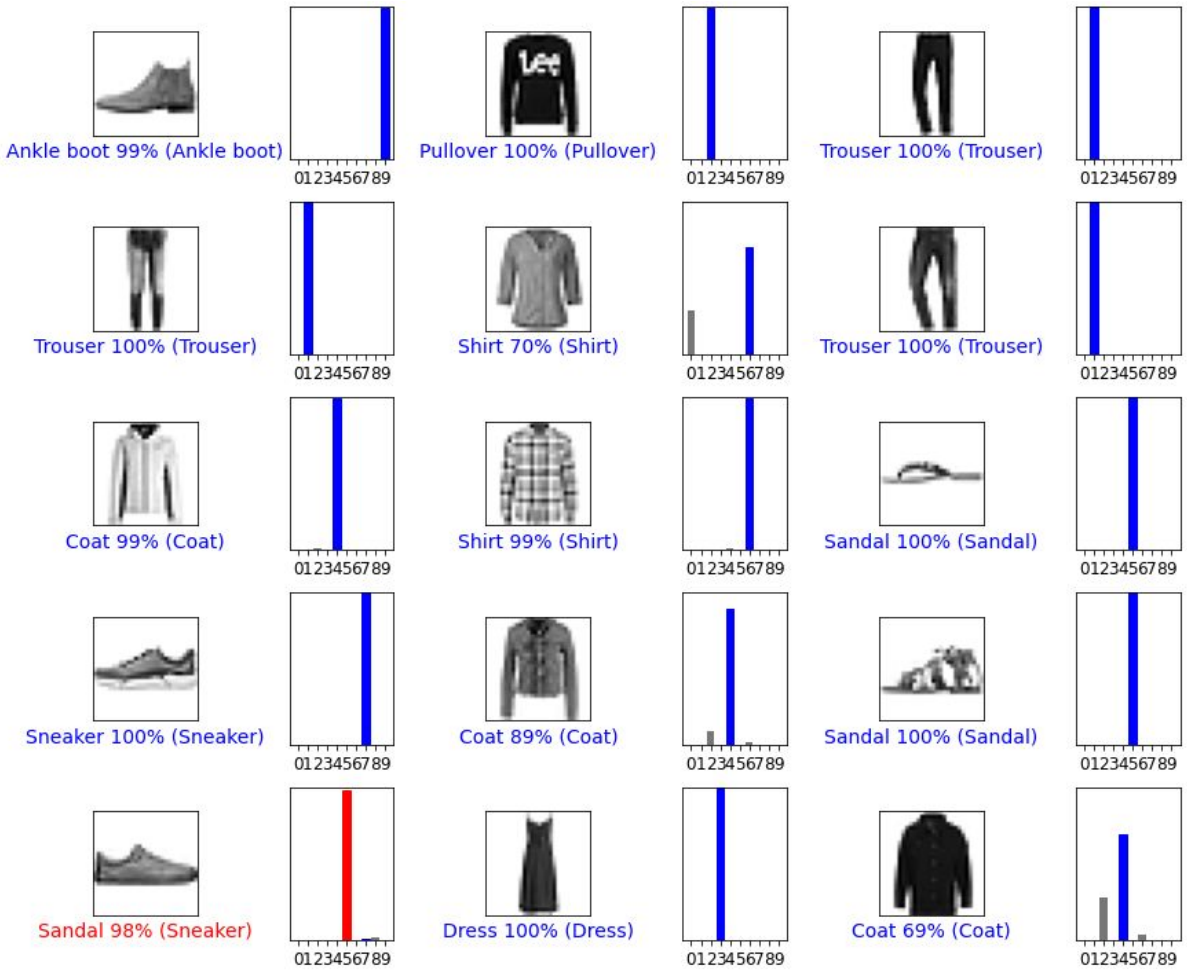
```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

train_images = train_images / 255.0

test_images = test_images / 255.0

model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)

print('\nTest accuracy:', test_acc)
```

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)

def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')

num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
```

```
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```

- Wynik:

```
2.4.0
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6175 - accuracy: 0.7873
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3888 - accuracy: 0.8600
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3426 - accuracy: 0.8755
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3167 - accuracy: 0.8846
Epoch 5/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2976 - accuracy: 0.8904
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2811 - accuracy: 0.8958
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2679 - accuracy: 0.9009
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2553 - accuracy: 0.9046
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2482 - accuracy: 0.9056
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2364 - accuracy: 0.9123
313/313 - 0s - loss: 0.3156 - accuracy: 0.8899

Test accuracy: 0.8899000287055969
```



Prognozy w postaci tablic predykcji
Etykiety prawidłowych prognoz są niebieskie, a nieprawidłowe etykiety prognoz są czerwone. Liczba podaje procent (ze 100) przewidywanej etykiety. Warto zauważyć, że model może być błędny, nawet jeśli jest bardzo pewny. (przykład w lewym dolnym rogu)