Handout on Neural Networks

Applied Machine & Deep Learning (190.015)
Univ.-Prof. Dr. Elmar Rueckert[1]

[1], *rueckert@unileoben.ac.at*, *MontanuniversitätLeoben*, *Austria*
*October*3, 2023

This handout is used for deepening your knowledge in multi-layer-perceptrons.

1 Introduction to MLP

Go to the following web page: `https://playground.tensorflow.org`. You will see a web interface as shown in Figure 1.

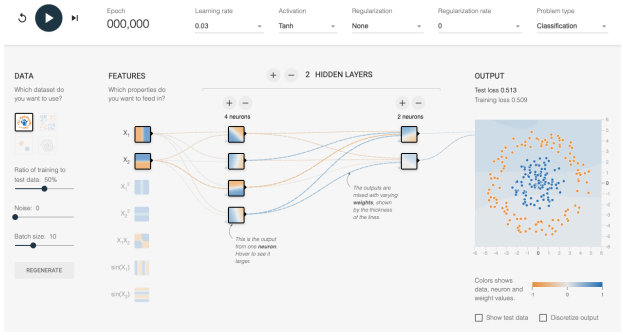

Figure 1: *Screenshot of the web interface for exploring multi-layer-perceptrons,* `https://playground.tensorflow.org`.

1.1 Understanding the interface

Explain what you can modify in the user interface. What are the individual elements? What happens if you move the mouse courser over individual neurons?

1.2 Simple three-layer network

Setup a simple MLP with three layers, two neurons in the hidden layer, two input neurons (x1 and x2). Try to solve the classification task using linear activation functions.

1.3 Limitations of Linear Activation Functions

Explain what you observed. Can the network solve the task? What changes are required to solve the task.

2 Increased model complexity

Change the task to the classification problem using the spiral dataset. Try to find an optimal configuration by adding hidden layers, neurons, inputs, changing the batch sizes, learning rate, etc.

2.1 Noise and batch sizes

Add noise to the datasets. Use at least a value of 30 or more. Train your network multiple times and reset the network to get a new random initialization of the weights (use the circular reset arrow just next to the play button). Ensure that the training converges.

What shape does each model output converge to? What does this say about the role of initialization in non-convex optimization? Does it always converge to an optimal solution? If not, explain why.

Come up with a strategy by changing the batch size to improve the model robustness with respect to noise in the dataset.

2.2 Noise and hidden layers

Now increase the model complexity by adding more neurons and more hidden layers. Train the model multiple times. Does it always converge to the optimal solution? If yes, explain why.

3 Regularization of MLPs

Explain how the regularization feature affects the model. Where is the regularization applied and how?

Which other types of regularization do you know in neural networks?

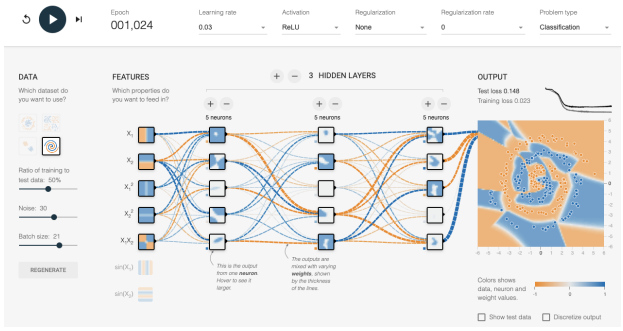

Figure 2: *Exploring multi-layer-perceptrons, a more complex example.*

3.1 Dropout using chatGPT

Implement a MLP with a dropout regularization using chatGPT. How does the regularization affect the learning convergence?

4 Literature

We strongly recommend to search for related tutorials, e.g., look at `https://www.baeldung.com/cs/ml-relu-dropout-layers`, `https://machinelearningknowledge.ai/keras-dropout-layer-explained-for-beginners/` or at `https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/`.

5 Conclusion

This tutorial is intended for undergraduate and graduate students to gain some first practical experience in using multi-layer perceptrons.

```python
# Import necessary libraries
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load the MNIST dataset (as an example)
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess the data
x_train = x_train.reshape(-1, 784) / 255.0
x_test = x_test.reshape(-1, 784) / 255.0
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Create a Sequential model
model = Sequential()

# Add input layer
model.add(Dense(units=512, activation='relu', input_dim=784))
model.add(Dropout(0.5))  # Add dropout layer with dropout rate of 0.5

# Add a hidden layer
model.add(Dense(units=256, activation='relu'))
model.add(Dropout(0.5))

# Add another hidden layer
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(0.5))
```

Figure 3: *Parts of the code generated using chatGPT implementing a MLP with dropout regularization.*

APPENDIX

Maintain a history of your chat queries and the responses when using large language tools like chatGPT. Add this history to any report or documentation, e.g., to the appendix.

Adding code

Adding code should be avoided. Preferably add links to Jupyter notebooks or git repositories.

However, in case you want to print code snippets, use markup libraries as shown here.