Ostbayerische Technische Hochschule
Amberg-Weiden

Neural networks exercise

Neural networks exercise

Ostbayerische Technische Hochschule
Amberg-Weiden

Exercise 1:

Create a neural network to learn the function

$$f(x) = \begin{bmatrix} \sin(x_1 x_2) \\ \cos(x_1 + x_2) \\ (x_1 x_2)^2 \end{bmatrix} \qquad \text{within the bounds} \quad x_1, x_2 \in [-2, 2]$$

Hints:

- Try out more than one hidden layer
- Experiment with different learning rates, optimizer (SGD vs. Adam), activation functions (Tanh vs. ReLU) and the size of the hidden layers

Neural networks exercise

Ostbayerische Technische Hochschule
Amberg-Weiden

Exercise 2:

Improve your neural network from the previous exercise:

- Plot train and test error: Create a second test dataset of the same function with different sampling points. Calculate the loss of the test dataset (but don't optimize on this dataset!) and plot it together with the loss of the train dataset
- Use minibatches: At each iteration, do not learn on the entire train dataset, but a random subset (e.g. 128 randomly selected samples $\longleftrightarrow$ batch size = 128)

Neural networks exercise

Ostbayerische Technische Hochschule
Amberg-Weiden

Exercise 3:

Try to approximate the policy for a given gridworld environment from the previous exercises environment with a neural network

- Create an environment and learn a policy using policy iteration / value iteration / SARSA / Q-learning
- Use the `env._obs_to_state()` function to create samples of the form
  ```
  x_train = (x, y)
  y_train = argmax(p_up, p_right, p_down, p_left)
  ```
- Learn the policy and visualize it also for non-integer values (e.g. `(x, y) = (1.1, 2.7)`

Note:

- Use a Softmax output layer and a NLLLoss function or no Softmax output layer and a CrossEntropyLoss function
- When using RL there is in general no test dataset available