Lab Assignment 3 - Learning rates, activation functions, and optimization algorithms

9/12/2025

Attempt 1

In Progress

NEXT UP: Submit assignment

Add comment

Unlimited Attempts Allowed

29/11/2025 to 9/12/2025

Details

Go through each task below and test to run the code to get familiar with it.

Make a short report (pdf format) that includes: (1) plots/diagrams generated during training and evaluation, (2) key console output such as model summaries and training logs (loss/accuracy per epoch). Since the labs are only graded Pass/Fail, the report can be very simple, just copy and paste your results and figures given when running your program.

Upload your code as .py file/s.

During the scheduled lab sessions, present your code and report to a teacher. The focus should be on your observations, interpreting the results (what changed, why it matters).

Tasks

1. Train a deep MLP on the MNIST dataset, you can load it using keras.datasets.mnist.load_data(). See if you can get over 98% precision. Try searching for the optimal learning rate (i.e., by growing the learning rate exponentially, plotting the error, and finding the point where the error shoots up), and plot learning curves using **TensorBoard**.

2. Create a neural net for Fashion MNIST with 100 hidden layers, using the sigmoid, ReLU, ELU, and SELU activation functions, and discuss the vanishing/exploding gradients problem using each of the mentioned activation functions.

3. Practice training a deep neural network on the CIFAR10 image dataset. You can load it with keras.datasets.cifar10.load_data(). The dataset is composed of 60,000 32 × 32–pixel color images (50,000 for training, 10,000 for testing) with 10 classes, so you'll need a **softmax** output layer with 10 neurons.

- Build a DNN with 20 hidden layers of 100 neurons each (that's too many, but it's the point of this exercise). Use He initialization and the ELU activation function.
- Using Nadam optimization and early stopping, train the network and remember to search for the right learning rate each time you change the model's architecture or hyperparameters.
- Now try adding Batch Normalization and compare the learning curves. Is it converging faster than before? How does it affect training speed?
- Use regular Gradient Descent optimization and then substitute it with momentum optimization, Nesterov Accelerated Gradient, AdaGrad, RMSProp, Adam and Nadam. Discuss the difference.

Sample code for this assignment can be found at:

- Chapter 10: **https://github.com/ageron/handson-ml2/blob/master/10_neural_nets_with_keras.ipynb** (https://github.com/ageron/handson-ml2/blob/master/10_neural_nets_with_keras.ipynb)
- Chapter 11: **https://github.com/ageron/handson-ml2/blob/master/11_training_deep_neural_networks.ipynb** (https://github.com/ageron/handson-ml2/blob/master/11_training_deep_neural_networks.ipynb)

Choose a submission type

Web URL Upload Canvas Studio Office 365

‹
(https://hv.instructure.com/courses/10339/modules/items/460074)

S
assignment

›
(https://hv.instructure.com/courses/10339/modules/items/460