EE514 Assignment 2022/2023

There are two parts to this assignment. The first part requires making predictions about human actions from sensors worn on the body and will focus mainly on things like data wrangling and the application of classic statistical machine learning techniques. The second part focuses on a different problem: that of detecting the presence or absence of certain features from Mars imagery, and students will be required to use deep learning based approaches such as convolutional neural networks. For both the first and second part, students are provided with template Jupyter notebook files to get them started.

Compute platforms

Fitting deep neural networks is computationally expensive and is greatly accelerated by the use of GPU hardware. Most modern deep learning frameworks support acceleration via NVIDIA's CUDA toolkit. If you have a desktop or laptop with a high-end NVIDIA graphics card that has CUDA support and sufficient on-board memory, you should be able to train deep networks on your own machine.

Unfortunately, the GPUs typically used in deep learning research are quite expensive. Fortunately, there are some excellent online platforms that offer access to NVIDIA GPU hardware for free for small projects and demonstration purposes. Two such environments are Google Colab and Gradient. These platforms allow you to interactively edit and execute Jupyter notebooks on cloud GPUs and support all the most popular deep learning frameworks (e.g. PyTorch and TensorFlow).

Submission

The final submission should be a report documenting all assumptions, design decisions, and findings. Include visualizations, plots, and tables. You should strive to make your work completely reproducible using only the report document: include details on everything you tested and all results. Document and justify all design decisions. Also submit the notebooks with your best model for each part.

Important: do NOT include code as images (e.g. screenshots of code) in your report. Include code snippets as text.

Plagiarism

Please read and strictly adhere to the DCU Academic Integrity and Plagiarism Policy.

Note that reports are automatically checked against each other and against external web sources for plagiarism. Any suspected plagiarism will be treated seriously and may result in penalties and a zero grade (see Sec 6.2 of the DCU Academic Integrity and Plagiarism Policy).

Grading

The assignment is worth 25% of the overall mark for the module. Marks will be awarded based on the quality of the resulting report. In particular, I will be checking to see if you are handling data correctly, carrying out exploratory analysis to gain insights, correctly performing model selection, and critically, documenting everything in a clear and concise way. The submitted code will also be checked to ensure that the work is your own.

Part 1: Extrasensory

This part of the assignment focuses on using sensor data to predict human activity and is based on the ExtraSensory dataset, created by Ph.D. students and staff at the Department of Electrical and Computer Engineering, University of California, San Diego. You can read more about the dataset on their website.

Starter code

Download the starter code notebook. Following the instructions in the notebook to download the ExtraSensory dataset. Run the notebook (either on your own machine or on the cloud) to train and test a simple predictive model. You can use this as a starting point for the rest this part of the assignment.

Improving the test set

Select five different users and test the model on each individually. Evaluate the mean and variance of the test metrics (e.g., accuracy) across the five users. Combine the users into a single test set to reduce the variance in test metrics.

Validation data

There is no model selection in the provided starter code. Instead, the code simply trains on one user's data and tests on another. Introduce a validation set that is a combination of several users.

Increased training data

Investigate what happens when you change the user that is used for training by evaluating the results on the appropriate data split. Combine multiple users into a larger training set and evaluate the results on the appropriate data split.

Model selection

The C parameter of the logistic regression modulates the strength of the regularization applied to the model. Investigate what happens as this is increased or reduced. Experiment with additional machine learning algorithms from the scikit-learn package and compare the results. Also tune the hyperparameters of these models.

Model testing

Evaluate the chosen model on the held-out test set. Estimate OOS error and evaluate various other metrics from the sklearn.metrics package, such as balanced accuracy, precision, recall, F1, and ROC-AUC. Analyse and interpret the meaning of the results and comment on how well you believe the model you have created would work in the real world. Plot an ROC curve and comment on it.

Predicting other actions

Extend the previous work to predict other actions apart from walking, such as sitting or bicycling. Introduce additional features as needed.

Part 2: Planet Four

In this part of the assignment you will experiment with using deep convolutional neural networks to identify the presence of features on the surface of Mars from imagery. There are two features of interest: fans and blotches. The dataset you will use is a low-resolution version of the data annotated in the Planet Four citizen science project.

Download the dataset and starter code

Download and unpack the starter code notebook. Create a subfolder called "data" alongside the starter code notebook and download and unpack the dataset in this folder. Rename the dataset folder "splits". (Alternatively, edit the path in the `PlanetFourDataset` object to point to your dataset).

Train a model

Use the provided starter code to train a ResNet50 model for 5 epochs (passes through the dataset). You can do this either on your own machine or using Google Colab or Gradient. If you are using one of these platforms, you'll need to upload the notebook and data.

Notice after each epoch, the training routine will report validation loss and accuracy. Try experimenting with training for more epochs (note: calling `train()` again will train the current model for longer rather than training a new model from scratch). Plot the learning curves and comment on the accuracies achieved in the report. Modify the training routine to save a checkpoint of the model after each epoch.

Experiments

Experiment with training with different network architectures from torchvision.models to see if you can improve validation accuracy. Experiment with different learning rates and different optimizers such as Adam from torch.optim. Experiment with different data augmentation strategies from torchvision.transforms.

Evaluation

Update the notebook to evaluate the final selected model on the test set.