Regularization and Optuna

SYDE 599 Deep Learning F23

October 26, 2023

UNIVERSITY OF WATERLOO

Assignments

- If there are Assignment 1 questions, you can ask after class
- Self-enroll groups available for Assignment 2 (2-3 students) and Assignment 3 / Presentation (4-5 students)
- Assignment 2 should be easily completed with only 2 people, groups from Assignment 1 can split into two groups
- Completing A2 should help you feel ready to use PyTorch for the project

Optuna

- Not installed by default in Google Colab, install with `!pip install optuna`
- Optuna documentation -> Key features tutorials

UNIVERSITY OF WATERLOO

Key Optuna Terminology

- Study
  - Hyperparameter optimization session, consisting of a set of trials
- Trial
  - Process of evaluating an objective function once
  - Provides interface to get suggested parameters based on optimization algorithm
  - Considerations
    - Range of values
    - Log scale (e.g. learning rate, L2 penalty)
    - Discrete or continuous

`suggest_categorical` ()	Suggest a value for the categorical parameter.
`suggest_discrete_uniform` (name, low, high, q)	Suggest a value for the discrete parameter.
`suggest_float` (name, low, high, *[, step, log])	Suggest a value for the floating point parameter.
`suggest_int` (name, low, high[, step, log])	Suggest a value for the integer parameter.

Key Optuna Terminology

- Objective
  - Non-differentiable goal you want to optimize, e.g. maximize test accuracy on MNIST with a neural network
  - Single function, input is a “trial” and output is your metric
  - Suggest hyperparameters, construct model, perform training, and perform evaluation within the objective
- Sampler
  - Non-gradient based optimization method for (hyper)parameters
  - Consider reporting reasons for sampler choice in the report

Optuna Objective for Deep Learning

```
def objective(trial):
    hps = suggest_hyperparameters(trial)
    model = create_model(hps)
    model = train_model(model, train_dataset, hps)
    metric = evaluate_model(model, test_dataset)
    return metric
```

Convolution in Torch

- Torch expects image tensors of shape (B, **C**, H, W)
- We have 5 parameters to worry about in convolution
  - `in_channels, out_channels`: How will the feature/channel dimension change?
  - `kernel_size`: Kernel is shape `(k, k)` typically
  - `padding`: How to deal with the edges? We almost always use “same” padding (or `k//2`) to have a better understanding of how shapes of tensors will change through the network
  - `stride`: Do we downsample by factor of `s`?

Shapes in Convolution

- Channels
  - (B, C_in, H, W) -> (B, C_out, H, W)
- Padding, kernel size
  - If we use `"same"`, then H and W don't change
- Stride
  - (B, C, H, W) -> (B, C, H//s, W//s)
  - Keyword "same" doesn't work, must use `padding=k//2` to ensure shapes change as expected
  - We typically downsample at certain points in the network, not often a hyperparameter

Regularization Activity

- Given this network that overfits, what should be done to improve its test performance?

Convolution Shapes Activity

- Work through how convolution affects input/output shapes (time permitting)