

Homework 4: Deep Averaging Networks

Learning Objectives

- Understand feed-forward networks for classification
 - By implementing the DAN
- Develop understanding of an adaptive optimizer (Adagrad)
- Test out various regularization techniques [L2, word dropout]

1: Implement the DAN

- In data.py:
 - Generate bag of words representation for one example
- In model.py:
 - Implement DeepAveragingNetwork.forward
 - Example from edugrad/examples/toy_half_sum:
- In ops.py:
 - Implement exp Operation
 - softmax_rows
 - cross_entropy_loss

```
class MLP(nn.Module):
    def __init__(self, input_size, output_size):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, 32)
        self.fc2 = nn.Linear(32, 32)
        self.output = nn.Linear(32, output_size)

    def forward(self, inputs):
        hidden = edugrad.ops.relu(self.fc1(inputs))
        hidden = edugrad.ops.relu(self.fc2(hidden))
        return self.output(hidden)
```

2: Implement Adagrad Optimizer

- In `optim.py`: implement `Adagrad.step`
 - `param._grad_hist` should store the sum of squared gradients throughout training
 - You need to update this
 - Compute the adjusted learning rate
 - Update parameters
- Example, `edugrad.optim.SGD`:

```
class SGD(Optimizer):
    def __init__(self, params: Iterable[Tensor], lr=1e-4):
        super(SGD, self).__init__(params)
        self.lr = lr

    def step(self):
        for param in self.params:
            param.value -= self.lr * param.grad
        self._cur_step += 1
```

3: Train some DANs!

- run.py has a basic training loop for a DAN on SST data. For each run, it records (and you need to report):
 - Per epoch training loss, dev loss
 - Final model dev accuracy
- Three runs:
 - Default arguments
 - Plus L2 regularization
 - Plus L2 regularization and word dropout
- We will ask you to describe what trends you see in each run, and across runs.