DAML walkthrough

The main point of interest is `train_maml` in model.py.

prev_min_loss gets set to a high value on line 120 using bit shifting.

Then we go through epochs. This corresponds to train() in metatrainer.py.

Self.training_adjust() on line 127 just returns? Weird self.m.self_adjust() on line 128 is referring to the TSD model in tsd_net.py.

This also just passes? WTH?

line 132 handles the data reading and dumps the result in `turn_batches_domain`.

Lines 134 and 135 declare two optimizers, optim, and meta_optim. They use a filter expression `meta_optim = Adam(lr = self.meta_lr, params=filter(lambda x: x.requires_grad, self.m.parameters()),` .

This obtains only the parameters that need gradient updates. I'm not currently doing this step, so this is potentially where my stuff is going awry. I am doing a similar filter on line 241. However, this is filtering the parameters before the model copy happens, not when the optimizer is created.

They then copy the model's state dict using copy.deepcopy. I think this is the python internal deepcopy mechanism instead of the pytorch specific one.

For batch in data, for each task get the data.

Load the state dict into the model, zero out the graidents on optim (I think this is the inner optimizer)

They call self._convert_batch() on line 154. Need to look and see what that does but my initial reaction is that it's unimportant and specific to their task setup.

For each step in cfg.maml_steps run the input through the model and get the loss, pr_loss, m_loss and turn_states

call loss.backward() to get gradients. grab the model parameters and gradient clip them Then step

the inner optimizer (optim)

Once out of that loop, run the model again. It says it's using fresh data in the comment on line 178 but as far as I can tell, it is not.

record the losses in an array.

Then outside that loop, identify

Revision #4
Created Tue, Feb 11, 2020 5:28 PM by kenneth
Updated Tue, Feb 11, 2020 7:10 PM by kenneth