ckkissane first commit 5bde1d5 · 4 months ago History

Code Blame 82 lines (72 loc) · 2.83 KB Raw

```python
from utils import *
from crosscoder import CrossCoder
from buffer import Buffer
import tqdm

from torch.nn.utils import clip_grad_norm_
class Trainer:
    def __init__(self, cfg, model_A, model_B, all_tokens):
        self.cfg = cfg
        self.model_A = model_A
        self.model_B = model_B
        self.crosscoder = CrossCoder(cfg)
        self.buffer = Buffer(cfg, model_A, model_B, all_tokens)
        self.total_steps = cfg["num_tokens"] // cfg["batch_size"]

        self.optimizer = torch.optim.Adam(
            self.crosscoder.parameters(),
            lr=cfg["lr"],
            betas=(cfg["beta1"], cfg["beta2"]),
        )
        self.scheduler = torch.optim.lr_scheduler.LambdaLR(
            self.optimizer, self.lr_lambda
        )
        self.step_counter = 0

```

Symbols

Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.

Filter symbols r

- class Trainer
  - func __init__
  - func lr_lambda
  - func get_l1_coeff
  - func step
  - func log
  - func save
  - func train

```
        wandb.init(project=cfg["wandb_project"], entity=cfg["wandb_entity"])

    def lr_lambda(self, step):
        if step < 0.8 * self.total_steps:
            return 1.0
        else:
            return 1.0 - (step - 0.8 * self.total_steps) / (0.2 * self.total_steps)

    def get_l1_coeff(self):
        # Linearly increases from 0 to cfg["l1_coeff"] over the first 0.05 * self.t
        if self.step_counter < 0.05 * self.total_steps:
            return self.cfg["l1_coeff"] * self.step_counter / (0.05 * self.total_st
        else:
            return self.cfg["l1_coeff"]

    def step(self):
        acts = self.buffer.next()
        losses = self.crosscoder.get_losses(acts)
        loss = losses.l2_loss + self.get_l1_coeff() * losses.l1_loss
        loss.backward()
        clip_grad_norm_(self.crosscoder.parameters(), max_norm=1.0)
        self.optimizer.step()
        self.scheduler.step()
        self.optimizer.zero_grad()

        loss_dict = {
            "loss": loss.item(),
            "l2_loss": losses.l2_loss.item(),
            "l1_loss": losses.l1_loss.item(),
            "l0_loss": losses.l0_loss.item(),
            "l1_coeff": self.get_l1_coeff(),
            "lr": self.scheduler.get_last_lr()[0],
            "explained_variance": losses.explained_variance.mean().item(),
            "explained_variance_A": losses.explained_variance_A.mean().item(),
            "explained_variance_B": losses.explained_variance_B.mean().item(),
        }
        self.step_counter += 1
        return loss_dict

```

```
    def log(self, loss_dict):
        wandb.log(loss_dict, step=self.step_counter)
        print(loss_dict)

    def save(self):
        self.crosscoder.save()

    def train(self):
        self.step_counter = 0
        try:
            for i in tqdm.trange(self.total_steps):
                loss_dict = self.step()
                if i % self.cfg["log_every"] == 0:
                    self.log(loss_dict)
                if (i + 1) % self.cfg["save_every"] == 0:
                    self.save()
        finally:
            self.save()
```