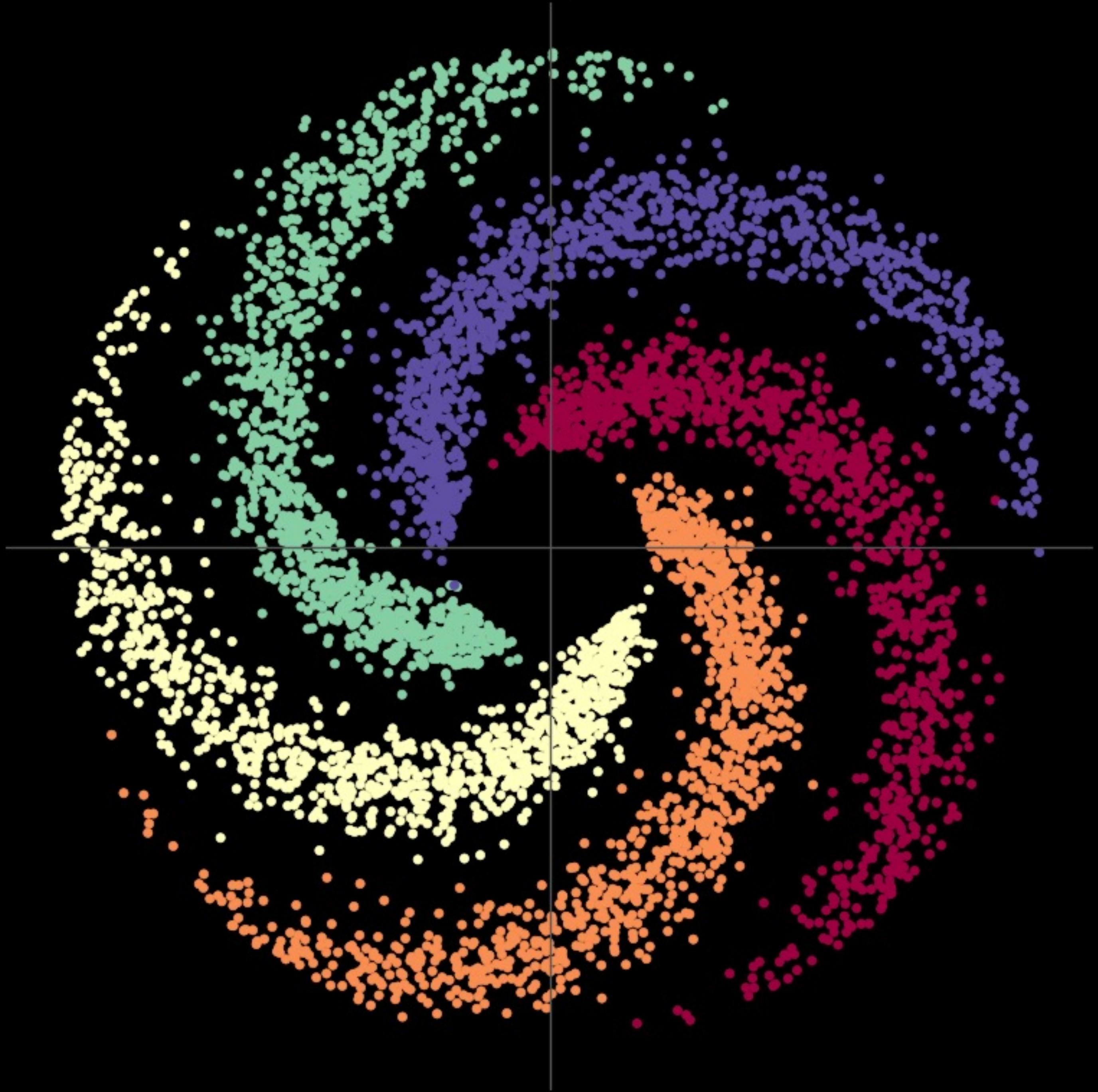
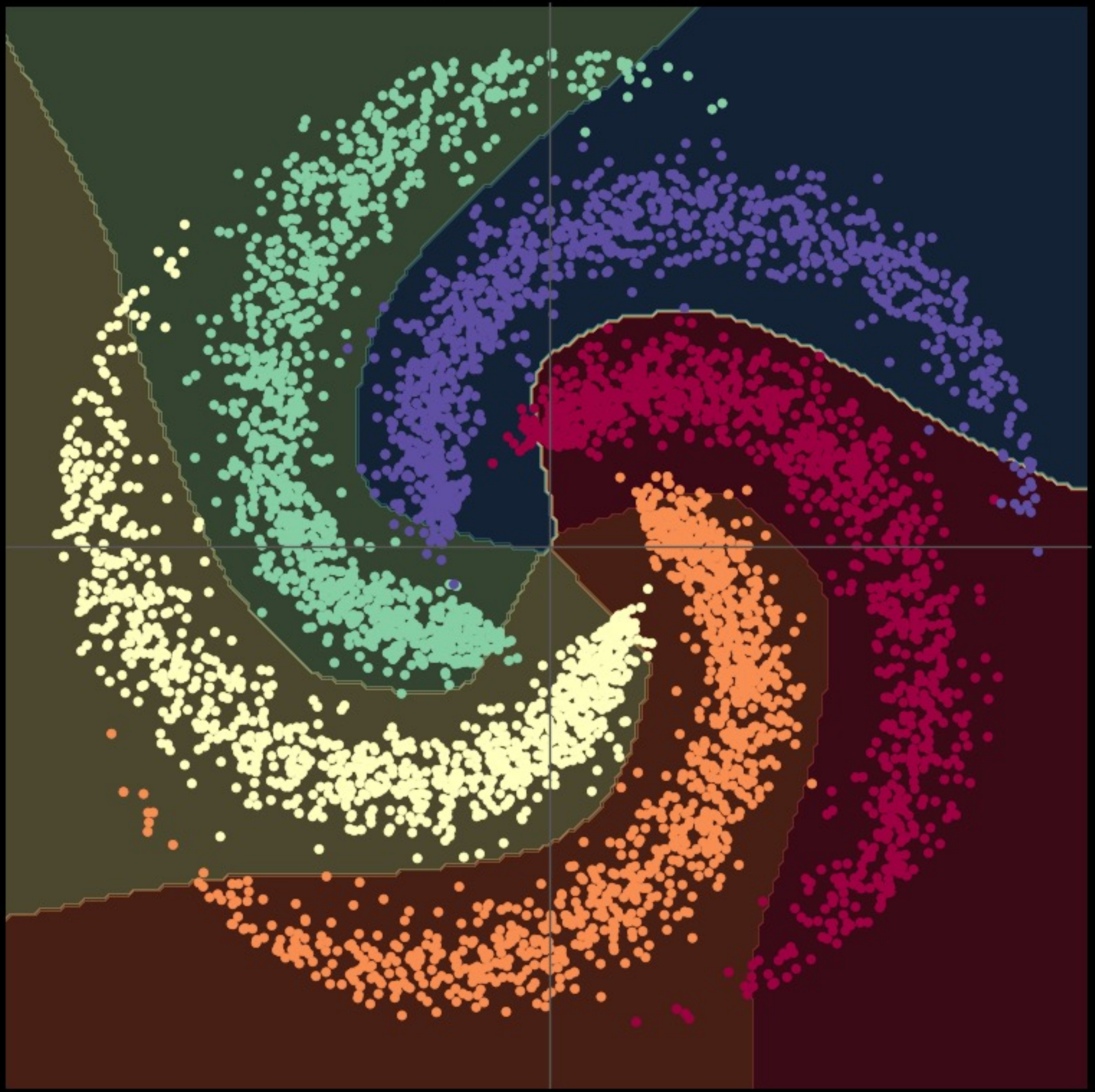
Free energy

Visuals

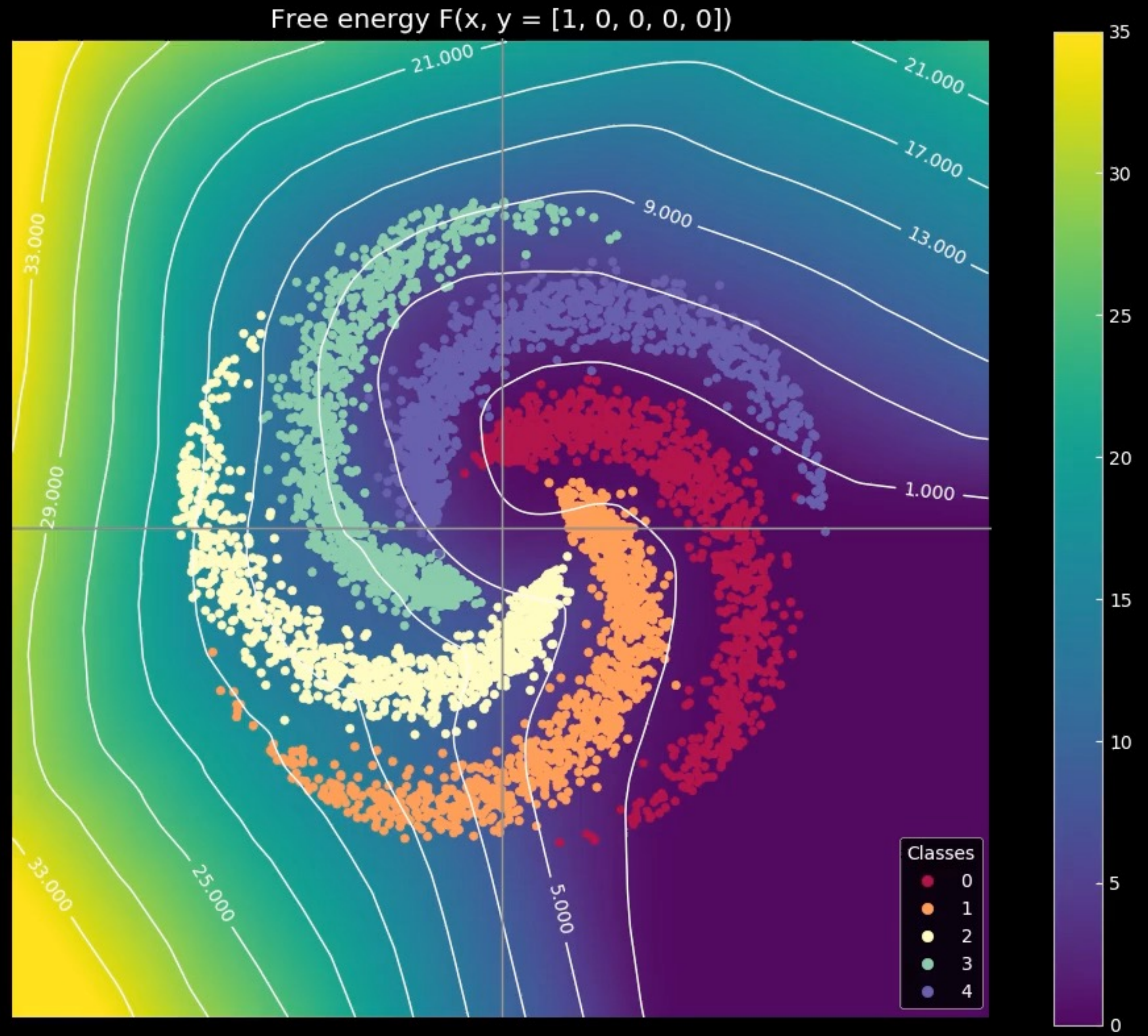
Training data (x, y)



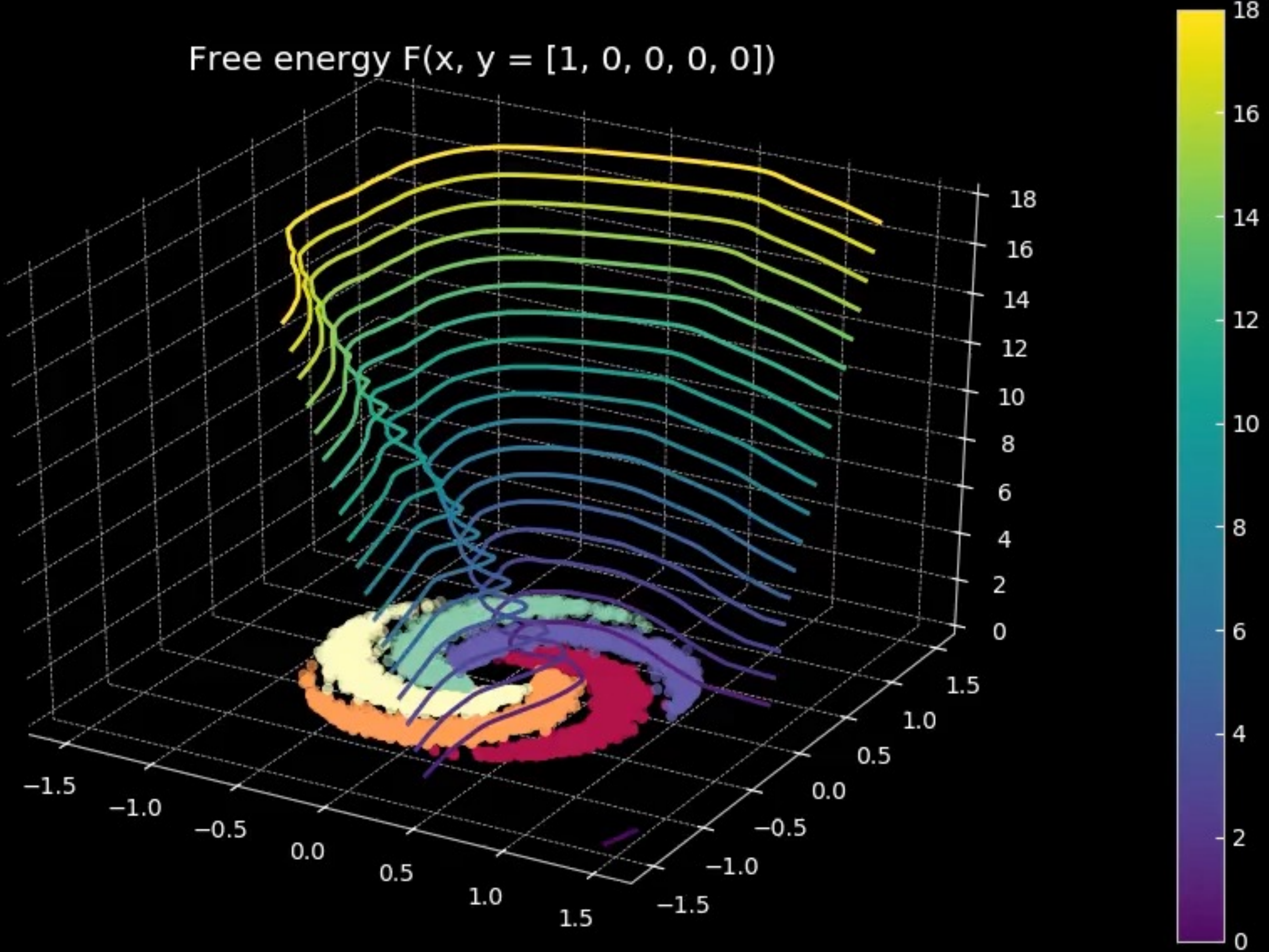
Model decision boundaries



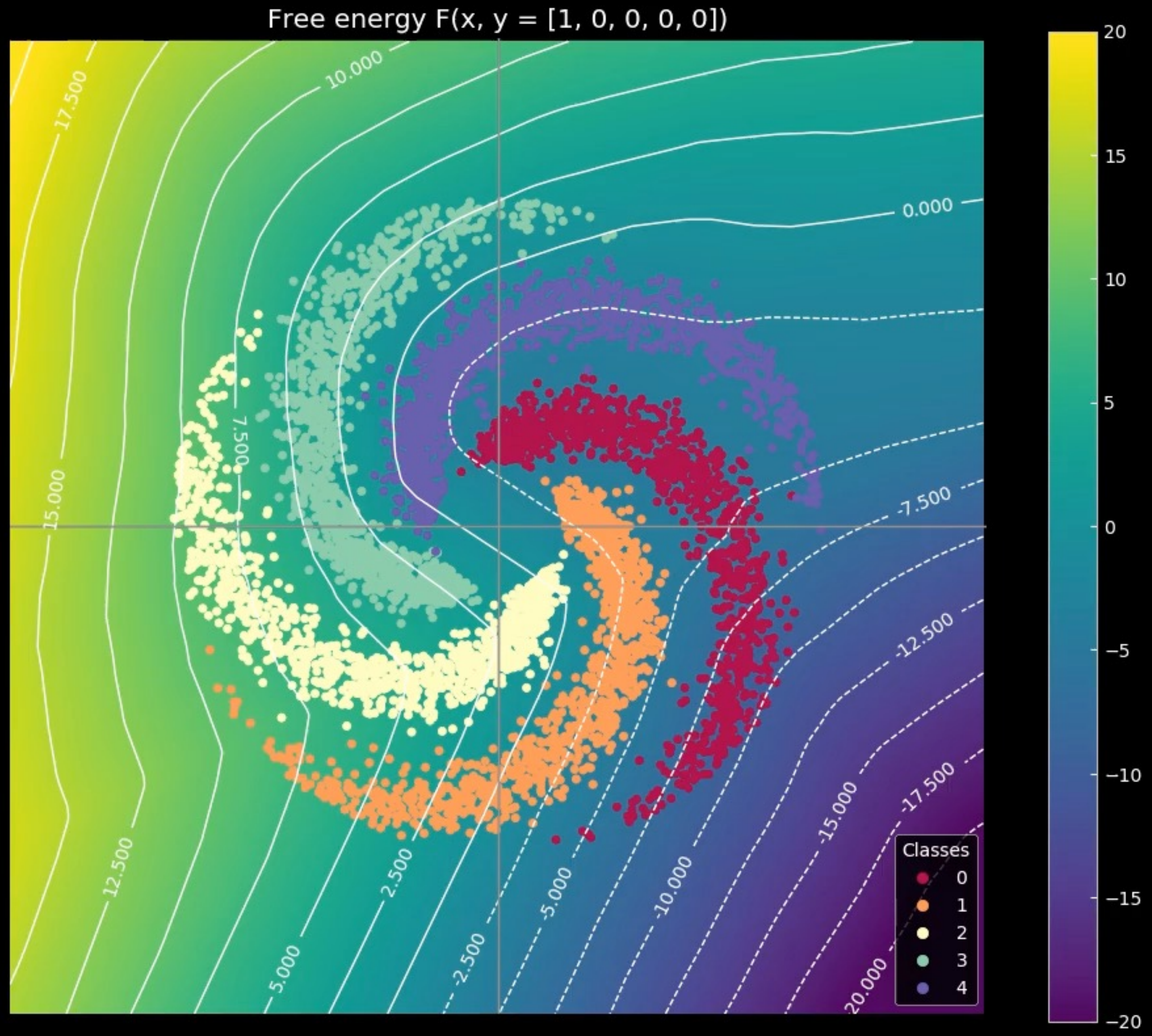
Cross-entropy free energy



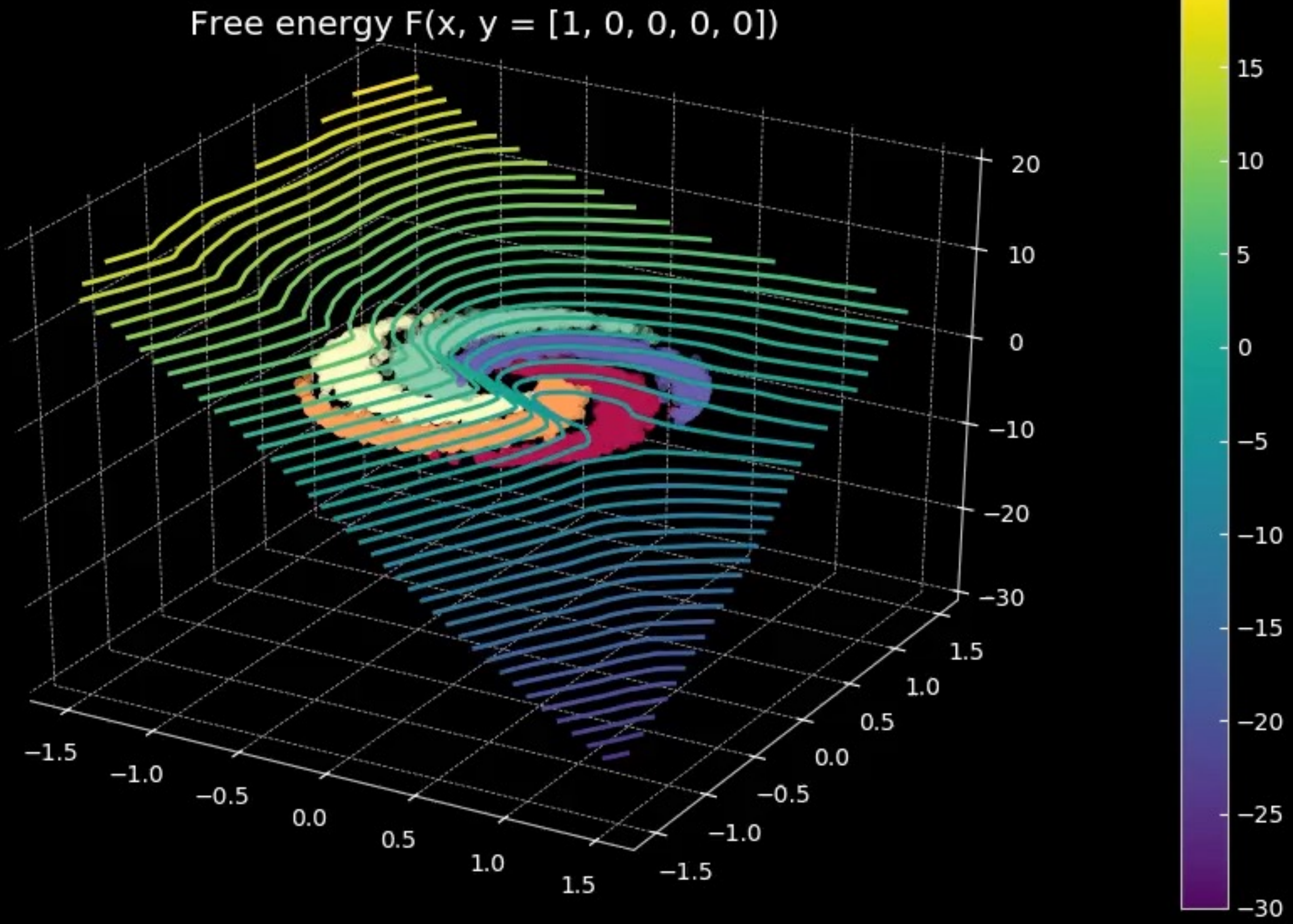
Cross-entropy free energy



Negative linear output free energy



Negative linear output free energy



PyTorch

Training a classifier

Setting the environment

- `import torch`
- `from torch import nn, optim`

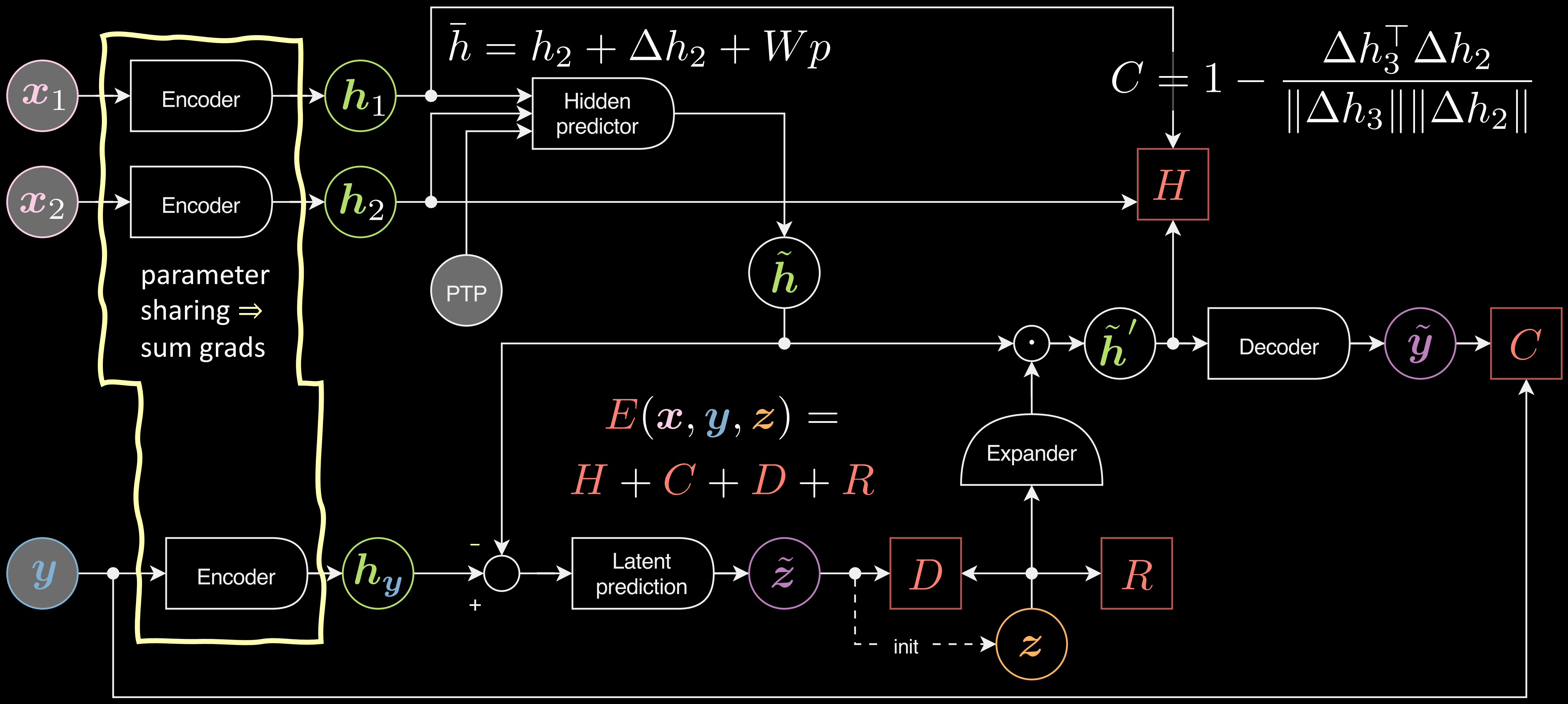
- `device = torch.device(…)  # CPU or GPU or TPU`
- `model = nn.Sequential(…).to(device)`
- `C = nn.CrossEntropyLoss() # cost definition`
- `optimiser = optim.SGD(model.parameters())`

Training loop 5 steps

```
for (x, y) in dataset:

1.  ỹ = model(x)  # generate a prediction
2.  L = F = C(ỹ, y)  # compute the loss
3.  optimiser.zero_grad()  # zero ∇params
4.  L.backward()  # compute & accumulate ∇params
5.  optimiser.step()  # step in towards -∇params
                             # logging
    …
```

Why backward accumulates ∇params (I)



Why backward accumulates ∇params (II)

```
optimiser.zero_grad()
ỹ₁ = model(x₁)
L = F = C(ỹ₁, y₁)
L.backward()  # compute & accumulate ∇params₁
ỹ₂ = model(x₂)
L = F = C(ỹ₂, y₂)
L.backward()  # compute & accumulate ∇params₂
optimiser.step()  # step towards -(∇₁ + ∇₂)
```