# 10

January 23, 2024

[ ]:
```
%pylab inline
import torch
from torch.nn.parameter import Parameter
```

[ ]:
```
x = torch.rand([1000,2])
x_in_circle = ((x**2).sum(1) < 1)

def accuracy(pred_label):
    return (pred_label==x_in_circle).float().mean()

def show(pred_label):
    scatter(*x.numpy().T, c=pred_label.numpy())
    axis('equal')

def loss(prediction):
    return -(x_in_circle.float()     * (prediction+1e-10).log() +
             (1-x_in_circle.float()) * (1-prediction+1e-10).log() ).mean()

class Linear(torch.nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.w = Parameter(torch.zeros(input_dim))
        self.b = Parameter(-torch.zeros(1))

    def forward(self, x):
        return (x * self.w[None,:]).sum(dim=1) + self.b

class LinearClassifier(torch.nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.linear = Linear(input_dim)

    def forward(self, x):
        logit = self.linear(x)
        return 1/(1+(-logit).exp())

show(x_in_circle)
```

[ ]:
```
import torch.utils.tensorboard as tb
%load_ext tensorboard
import tempfile
log_dir = tempfile.mkdtemp()
%tensorboard --logdir {log_dir} --reload_interval 1
```

[ ]:
```
logger = tb.SummaryWriter(log_dir+'/linear1')

classifier = LinearClassifier(2)

for iteration in range(10000):
    p_y = classifier(x)
    pred_y = p_y > 0.5
    l = loss(p_y)

    logger.add_scalar("loss", l, global_step=iteration)
    logger.add_scalar("accuracy", accuracy(pred_y), global_step=iteration)

    if iteration % 100 == 0:
        fig = figure()
        show(pred_y)
        logger.add_figure('pred_y', fig, global_step=iteration)
        del fig

    l.backward()
    for p in classifier.parameters():
        p.data[:] -= 0.5 * p.grad
        p.grad.zero_()

show(pred_y)
```

[ ]:
```
class NonLinearClassifier(torch.nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.linear1 = torch.nn.Linear(input_dim, 100)
        torch.nn.init.normal_(self.linear1.weight, std=0.01)
        torch.nn.init.normal_(self.linear1.bias, std=0.01)
        self.linear2 = Linear(100)

    def forward(self, x):
        logit = self.linear2( torch.relu(self.linear1(x)) )
        return 1/(1+(-logit).exp())


classifier = NonLinearClassifier(2)
show(classifier(x).detach() > 0.5)
```

[ ]:
```
logger = tb.SummaryWriter(log_dir+'/nonlinear1')

classifier = NonLinearClassifier(2)

for iteration in range(10000):
    p_y = classifier(x)
    pred_y = p_y > 0.5
    l = loss(p_y)

    logger.add_scalar("loss", l, global_step=iteration)
    logger.add_scalar("accuracy", accuracy(pred_y), global_step=iteration)

    if iteration % 100 == 0:
        fig = figure()
        show(pred_y)
        logger.add_figure('pred_y', fig, global_step=iteration)
        del fig

    l.backward()
    for p in classifier.parameters():
        p.data[:] -= 0.5 * p.grad
        p.grad.zero_()

show(pred_y)
```

[ ]:
```

```