SGD vs ADAM vs L-BFGS

May 5, 2022

[1]:
```
import torch
import torch.nn as nn
from torch.autograd import Variable
from torch.optim import Adam, LBFGS, SGD
from torch.utils.data import Dataset, DataLoader

class LinearModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(1, 1)

    def forward(self, x):
        return self.linear(x)
```

[2]:
```
x = torch.linspace(-5, 5, steps=20)
y = 2 * x + 2
x, y = x.reshape(-1, 1), y.reshape(-1, 1)
```

[3]:
```
class MyData(Dataset):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __len__(self):
        return self.x.shape[0]

    def __getitem__(self, idx):
        return self.x[idx], self.y[idx]
```

[4]:
```
# Determine if it is a cpu or a gpu
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# DataSet
my_data = MyData(x.to(device), y.to(device))

# Training parameters
criterion = nn.MSELoss()
epochs = 100
```

[5]:
```
lm_sgd = LinearModel()
lm_sgd.to(device)
optimizer = SGD(lm_sgd.parameters(), lr=0.0001)
sgd_loss = []
for epoch in range(epochs):
    running_loss = 0.0

    for i, (x, y) in enumerate(my_data):

        x_ = Variable(x, requires_grad=True)
        y_ = Variable(y)

        # Forward pass
        y_pred = lm_sgd(x_)

        # Compute loss
        loss = criterion(y_pred, y_)

        # Zero gradients, backward pass, and update weights
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # Update the running loss
        running_loss += loss.item()
    sgd_loss.append(running_loss)
    print(f"Epoch: {epoch + 1:02}/{epochs} Loss: {running_loss:.5e}")
```

```
Epoch: 01/100 Loss: 1.62432e+03
Epoch: 02/100 Loss: 1.51894e+03
Epoch: 03/100 Loss: 1.42098e+03
Epoch: 04/100 Loss: 1.32992e+03
Epoch: 05/100 Loss: 1.24527e+03
Epoch: 06/100 Loss: 1.16656e+03
Epoch: 07/100 Loss: 1.09339e+03
Epoch: 08/100 Loss: 1.02534e+03
Epoch: 09/100 Loss: 9.62067e+02
Epoch: 10/100 Loss: 9.03221e+02
Epoch: 11/100 Loss: 8.48491e+02
Epoch: 12/100 Loss: 7.97583e+02
Epoch: 13/100 Loss: 7.50227e+02
Epoch: 14/100 Loss: 7.06169e+02
Epoch: 15/100 Loss: 6.65176e+02
Epoch: 16/100 Loss: 6.27030e+02
Epoch: 17/100 Loss: 5.91528e+02
Epoch: 18/100 Loss: 5.58483e+02
Epoch: 19/100 Loss: 5.27721e+02
Epoch: 20/100 Loss: 4.99079e+02
```

```
Epoch: 21/100 Loss: 4.72408e+02
Epoch: 22/100 Loss: 4.47566e+02
Epoch: 23/100 Loss: 4.24425e+02
Epoch: 24/100 Loss: 4.02864e+02
Epoch: 25/100 Loss: 3.82771e+02
Epoch: 26/100 Loss: 3.64042e+02
Epoch: 27/100 Loss: 3.46580e+02
Epoch: 28/100 Loss: 3.30295e+02
Epoch: 29/100 Loss: 3.15105e+02
Epoch: 30/100 Loss: 3.00931e+02
Epoch: 31/100 Loss: 2.87702e+02
Epoch: 32/100 Loss: 2.75351e+02
Epoch: 33/100 Loss: 2.63816e+02
Epoch: 34/100 Loss: 2.53040e+02
Epoch: 35/100 Loss: 2.42968e+02
Epoch: 36/100 Loss: 2.33551e+02
Epoch: 37/100 Loss: 2.24743e+02
Epoch: 38/100 Loss: 2.16501e+02
Epoch: 39/100 Loss: 2.08785e+02
Epoch: 40/100 Loss: 2.01558e+02
Epoch: 41/100 Loss: 1.94786e+02
Epoch: 42/100 Loss: 1.88437e+02
Epoch: 43/100 Loss: 1.82481e+02
Epoch: 44/100 Loss: 1.76890e+02
Epoch: 45/100 Loss: 1.71640e+02
Epoch: 46/100 Loss: 1.66705e+02
Epoch: 47/100 Loss: 1.62065e+02
Epoch: 48/100 Loss: 1.57699e+02
Epoch: 49/100 Loss: 1.53586e+02
Epoch: 50/100 Loss: 1.49711e+02
Epoch: 51/100 Loss: 1.46056e+02
Epoch: 52/100 Loss: 1.42606e+02
Epoch: 53/100 Loss: 1.39348e+02
Epoch: 54/100 Loss: 1.36266e+02
Epoch: 55/100 Loss: 1.33351e+02
Epoch: 56/100 Loss: 1.30589e+02
Epoch: 57/100 Loss: 1.27971e+02
Epoch: 58/100 Loss: 1.25487e+02
Epoch: 59/100 Loss: 1.23128e+02
Epoch: 60/100 Loss: 1.20884e+02
Epoch: 61/100 Loss: 1.18749e+02
Epoch: 62/100 Loss: 1.16714e+02
Epoch: 63/100 Loss: 1.14774e+02
Epoch: 64/100 Loss: 1.12922e+02
Epoch: 65/100 Loss: 1.11151e+02
Epoch: 66/100 Loss: 1.09457e+02
Epoch: 67/100 Loss: 1.07835e+02
Epoch: 68/100 Loss: 1.06279e+02
```

```
Epoch: 69/100 Loss: 1.04786e+02
Epoch: 70/100 Loss: 1.03351e+02
Epoch: 71/100 Loss: 1.01970e+02
Epoch: 72/100 Loss: 1.00640e+02
Epoch: 73/100 Loss: 9.93582e+01
Epoch: 74/100 Loss: 9.81209e+01
Epoch: 75/100 Loss: 9.69255e+01
Epoch: 76/100 Loss: 9.57692e+01
Epoch: 77/100 Loss: 9.46498e+01
Epoch: 78/100 Loss: 9.35649e+01
Epoch: 79/100 Loss: 9.25124e+01
Epoch: 80/100 Loss: 9.14904e+01
Epoch: 81/100 Loss: 9.04971e+01
Epoch: 82/100 Loss: 8.95306e+01
Epoch: 83/100 Loss: 8.85895e+01
Epoch: 84/100 Loss: 8.76723e+01
Epoch: 85/100 Loss: 8.67777e+01
Epoch: 86/100 Loss: 8.59043e+01
Epoch: 87/100 Loss: 8.50510e+01
Epoch: 88/100 Loss: 8.42166e+01
Epoch: 89/100 Loss: 8.34001e+01
Epoch: 90/100 Loss: 8.26007e+01
Epoch: 91/100 Loss: 8.18174e+01
Epoch: 92/100 Loss: 8.10493e+01
Epoch: 93/100 Loss: 8.02958e+01
Epoch: 94/100 Loss: 7.95560e+01
Epoch: 95/100 Loss: 7.88293e+01
Epoch: 96/100 Loss: 7.81152e+01
Epoch: 97/100 Loss: 7.74129e+01
Epoch: 98/100 Loss: 7.67220e+01
Epoch: 99/100 Loss: 7.60420e+01
Epoch: 100/100 Loss: 7.53724e+01
```

[6]:
```python
lm_adam = LinearModel()
lm_adam.to(device)
optimizer = Adam(lm_adam.parameters(), weight_decay=0.0001)
adam_loss = []
for epoch in range(epochs):
    running_loss = 0.0

    for i, (x, y) in enumerate(my_data):

        x_ = Variable(x, requires_grad=True)
        y_ = Variable(y)

        # Forward pass
        y_pred = lm_adam(x_)
```

```
        # Compute loss
        loss = criterion(y_pred, y_)

        # Zero gradients, backward pass, and update weights
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # Update the running loss
        running_loss += loss.item()
    adam_loss.append(running_loss)
    print(f"Epoch: {epoch + 1:02}/{epochs} Loss: {running_loss:.5e}")
```

```
Epoch: 01/100 Loss: 4.95932e+02
Epoch: 02/100 Loss: 4.86999e+02
Epoch: 03/100 Loss: 4.78552e+02
Epoch: 04/100 Loss: 4.70345e+02
Epoch: 05/100 Loss: 4.62325e+02
Epoch: 06/100 Loss: 4.54459e+02
Epoch: 07/100 Loss: 4.46731e+02
Epoch: 08/100 Loss: 4.39131e+02
Epoch: 09/100 Loss: 4.31651e+02
Epoch: 10/100 Loss: 4.24287e+02
Epoch: 11/100 Loss: 4.17035e+02
Epoch: 12/100 Loss: 4.09892e+02
Epoch: 13/100 Loss: 4.02855e+02
Epoch: 14/100 Loss: 3.95923e+02
Epoch: 15/100 Loss: 3.89092e+02
Epoch: 16/100 Loss: 3.82362e+02
Epoch: 17/100 Loss: 3.75731e+02
Epoch: 18/100 Loss: 3.69197e+02
Epoch: 19/100 Loss: 3.62759e+02
Epoch: 20/100 Loss: 3.56416e+02
Epoch: 21/100 Loss: 3.50166e+02
Epoch: 22/100 Loss: 3.44008e+02
Epoch: 23/100 Loss: 3.37941e+02
Epoch: 24/100 Loss: 3.31963e+02
Epoch: 25/100 Loss: 3.26074e+02
Epoch: 26/100 Loss: 3.20272e+02
Epoch: 27/100 Loss: 3.14557e+02
Epoch: 28/100 Loss: 3.08927e+02
Epoch: 29/100 Loss: 3.03381e+02
Epoch: 30/100 Loss: 2.97918e+02
Epoch: 31/100 Loss: 2.92537e+02
Epoch: 32/100 Loss: 2.87238e+02
Epoch: 33/100 Loss: 2.82019e+02
Epoch: 34/100 Loss: 2.76879e+02
```

```
Epoch: 35/100 Loss: 2.71818e+02
Epoch: 36/100 Loss: 2.66834e+02
Epoch: 37/100 Loss: 2.61926e+02
Epoch: 38/100 Loss: 2.57094e+02
Epoch: 39/100 Loss: 2.52336e+02
Epoch: 40/100 Loss: 2.47653e+02
Epoch: 41/100 Loss: 2.43042e+02
Epoch: 42/100 Loss: 2.38503e+02
Epoch: 43/100 Loss: 2.34035e+02
Epoch: 44/100 Loss: 2.29638e+02
Epoch: 45/100 Loss: 2.25310e+02
Epoch: 46/100 Loss: 2.21051e+02
Epoch: 47/100 Loss: 2.16859e+02
Epoch: 48/100 Loss: 2.12734e+02
Epoch: 49/100 Loss: 2.08676e+02
Epoch: 50/100 Loss: 2.04683e+02
Epoch: 51/100 Loss: 2.00754e+02
Epoch: 52/100 Loss: 1.96889e+02
Epoch: 53/100 Loss: 1.93087e+02
Epoch: 54/100 Loss: 1.89347e+02
Epoch: 55/100 Loss: 1.85668e+02
Epoch: 56/100 Loss: 1.82050e+02
Epoch: 57/100 Loss: 1.78492e+02
Epoch: 58/100 Loss: 1.74993e+02
Epoch: 59/100 Loss: 1.71553e+02
Epoch: 60/100 Loss: 1.68170e+02
Epoch: 61/100 Loss: 1.64844e+02
Epoch: 62/100 Loss: 1.61574e+02
Epoch: 63/100 Loss: 1.58359e+02
Epoch: 64/100 Loss: 1.55199e+02
Epoch: 65/100 Loss: 1.52093e+02
Epoch: 66/100 Loss: 1.49041e+02
Epoch: 67/100 Loss: 1.46041e+02
Epoch: 68/100 Loss: 1.43093e+02
Epoch: 69/100 Loss: 1.40196e+02
Epoch: 70/100 Loss: 1.37349e+02
Epoch: 71/100 Loss: 1.34553e+02
Epoch: 72/100 Loss: 1.31806e+02
Epoch: 73/100 Loss: 1.29107e+02
Epoch: 74/100 Loss: 1.26456e+02
Epoch: 75/100 Loss: 1.23852e+02
Epoch: 76/100 Loss: 1.21295e+02
Epoch: 77/100 Loss: 1.18784e+02
Epoch: 78/100 Loss: 1.16318e+02
Epoch: 79/100 Loss: 1.13897e+02
Epoch: 80/100 Loss: 1.11519e+02
Epoch: 81/100 Loss: 1.09185e+02
Epoch: 82/100 Loss: 1.06894e+02
```

```
Epoch: 83/100 Loss: 1.04645e+02
Epoch: 84/100 Loss: 1.02437e+02
Epoch: 85/100 Loss: 1.00271e+02
Epoch: 86/100 Loss: 9.81445e+01
Epoch: 87/100 Loss: 9.60580e+01
Epoch: 88/100 Loss: 9.40108e+01
Epoch: 89/100 Loss: 9.20020e+01
Epoch: 90/100 Loss: 9.00311e+01
Epoch: 91/100 Loss: 8.80978e+01
Epoch: 92/100 Loss: 8.62012e+01
Epoch: 93/100 Loss: 8.43410e+01
Epoch: 94/100 Loss: 8.25166e+01
Epoch: 95/100 Loss: 8.07273e+01
Epoch: 96/100 Loss: 7.89726e+01
Epoch: 97/100 Loss: 7.72520e+01
Epoch: 98/100 Loss: 7.55649e+01
Epoch: 99/100 Loss: 7.39109e+01
Epoch: 100/100 Loss: 7.22894e+01
```

[7]:
```python
lm_lbfgs = LinearModel()
lm_lbfgs.to(device)
optimizer = LBFGS(lm_lbfgs.parameters(), history_size=10, max_iter=4)
lbfgs_loss = []
for epoch in range(epochs):
    running_loss = 0.0

    for i, (x, y) in enumerate(my_data):

        x_ = Variable(x, requires_grad=True)
        y_ = Variable(y)

        def closure():
            # Zero gradients
            optimizer.zero_grad()

            # Forward pass
            y_pred = lm_lbfgs(x_)

            # Compute loss
            loss = criterion(y_pred, y_)

            # Backward pass
            loss.backward()

            return loss

        # Update weights
```

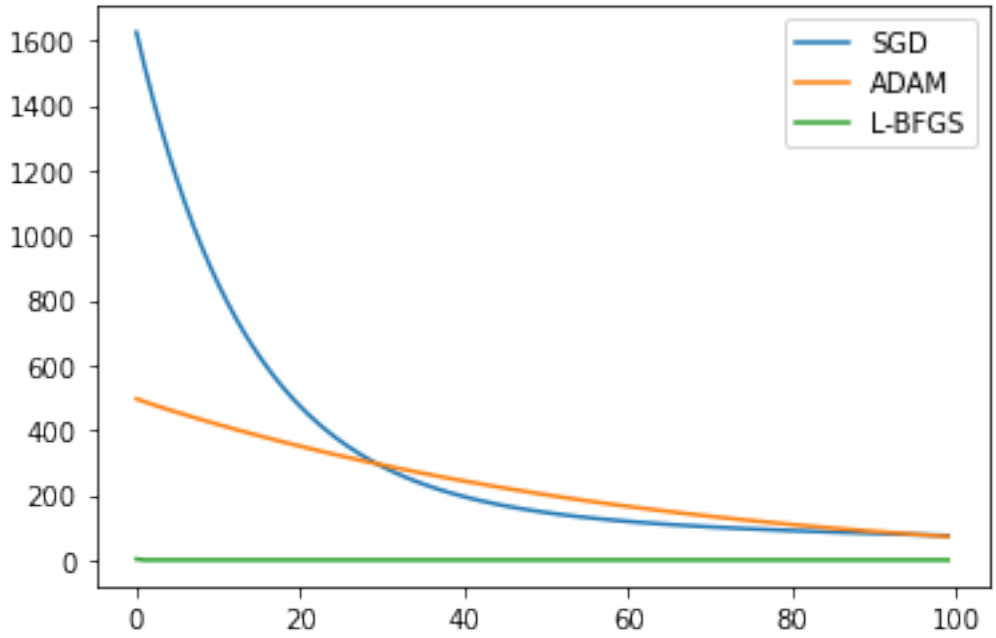
```
        optimizer.step(closure)

        # Update the running loss
        loss = closure()
        running_loss += loss.item()
    lbfgs_loss.append(running_loss)
    print(f"Epoch: {epoch + 1:02}/{epochs} Loss: {running_loss:.5e}")
```

```
Epoch: 01/100 Loss: 2.85742e+00
Epoch: 02/100 Loss: 5.55160e-04
Epoch: 03/100 Loss: 3.95409e-08
Epoch: 04/100 Loss: 3.93813e-08
Epoch: 05/100 Loss: 1.24924e-08
Epoch: 06/100 Loss: 8.16669e-10
Epoch: 07/100 Loss: 4.52872e-10
Epoch: 08/100 Loss: 4.52872e-10
Epoch: 09/100 Loss: 4.52872e-10
Epoch: 10/100 Loss: 4.52872e-10
Epoch: 11/100 Loss: 4.52872e-10
Epoch: 12/100 Loss: 4.52872e-10
Epoch: 13/100 Loss: 4.52872e-10
Epoch: 14/100 Loss: 4.52872e-10
Epoch: 15/100 Loss: 4.52872e-10
Epoch: 16/100 Loss: 4.52872e-10
Epoch: 17/100 Loss: 4.52872e-10
Epoch: 18/100 Loss: 4.52872e-10
Epoch: 19/100 Loss: 4.52872e-10
Epoch: 20/100 Loss: 4.52872e-10
Epoch: 21/100 Loss: 4.52872e-10
Epoch: 22/100 Loss: 4.52872e-10
Epoch: 23/100 Loss: 4.52872e-10
Epoch: 24/100 Loss: 4.52872e-10
Epoch: 25/100 Loss: 4.52872e-10
Epoch: 26/100 Loss: 4.52872e-10
Epoch: 27/100 Loss: 4.52872e-10
Epoch: 28/100 Loss: 4.52872e-10
Epoch: 29/100 Loss: 4.52872e-10
Epoch: 30/100 Loss: 4.52872e-10
Epoch: 31/100 Loss: 4.52872e-10
Epoch: 32/100 Loss: 4.52872e-10
Epoch: 33/100 Loss: 4.52872e-10
Epoch: 34/100 Loss: 4.52872e-10
Epoch: 35/100 Loss: 4.52872e-10
Epoch: 36/100 Loss: 4.52872e-10
Epoch: 37/100 Loss: 4.52872e-10
Epoch: 38/100 Loss: 4.52872e-10
Epoch: 39/100 Loss: 4.52872e-10
Epoch: 40/100 Loss: 4.52872e-10
```

```
Epoch: 41/100 Loss: 4.52872e-10
Epoch: 42/100 Loss: 4.52872e-10
Epoch: 43/100 Loss: 4.52872e-10
Epoch: 44/100 Loss: 4.52872e-10
Epoch: 45/100 Loss: 4.52872e-10
Epoch: 46/100 Loss: 4.52872e-10
Epoch: 47/100 Loss: 4.52872e-10
Epoch: 48/100 Loss: 4.52872e-10
Epoch: 49/100 Loss: 4.52872e-10
Epoch: 50/100 Loss: 4.52872e-10
Epoch: 51/100 Loss: 1.82752e-10
Epoch: 52/100 Loss: 6.29115e-11
Epoch: 53/100 Loss: 6.29115e-11
Epoch: 54/100 Loss: 6.29115e-11
Epoch: 55/100 Loss: 6.29115e-11
Epoch: 56/100 Loss: 6.29115e-11
Epoch: 57/100 Loss: 6.29115e-11
Epoch: 58/100 Loss: 6.29115e-11
Epoch: 59/100 Loss: 6.29115e-11
Epoch: 60/100 Loss: 6.29115e-11
Epoch: 61/100 Loss: 6.29115e-11
Epoch: 62/100 Loss: 6.29115e-11
Epoch: 63/100 Loss: 6.29115e-11
Epoch: 64/100 Loss: 6.29115e-11
Epoch: 65/100 Loss: 6.29115e-11
Epoch: 66/100 Loss: 6.29115e-11
Epoch: 67/100 Loss: 6.29115e-11
Epoch: 68/100 Loss: 6.29115e-11
Epoch: 69/100 Loss: 6.29115e-11
Epoch: 70/100 Loss: 6.29115e-11
Epoch: 71/100 Loss: 6.29115e-11
Epoch: 72/100 Loss: 6.29115e-11
Epoch: 73/100 Loss: 6.29115e-11
Epoch: 74/100 Loss: 6.29115e-11
Epoch: 75/100 Loss: 6.29115e-11
Epoch: 76/100 Loss: 6.29115e-11
Epoch: 77/100 Loss: 6.29115e-11
Epoch: 78/100 Loss: 6.29115e-11
Epoch: 79/100 Loss: 6.29115e-11
Epoch: 80/100 Loss: 6.29115e-11
Epoch: 81/100 Loss: 6.29115e-11
Epoch: 82/100 Loss: 6.29115e-11
Epoch: 83/100 Loss: 6.29115e-11
Epoch: 84/100 Loss: 6.29115e-11
Epoch: 85/100 Loss: 6.29115e-11
Epoch: 86/100 Loss: 6.29115e-11
Epoch: 87/100 Loss: 6.29115e-11
Epoch: 88/100 Loss: 6.29115e-11
```

```
Epoch: 89/100 Loss: 6.29115e-11
Epoch: 90/100 Loss: 6.29115e-11
Epoch: 91/100 Loss: 6.29115e-11
Epoch: 92/100 Loss: 6.29115e-11
Epoch: 93/100 Loss: 6.29115e-11
Epoch: 94/100 Loss: 6.29115e-11
Epoch: 95/100 Loss: 6.29115e-11
Epoch: 96/100 Loss: 6.29115e-11
Epoch: 97/100 Loss: 6.29115e-11
Epoch: 98/100 Loss: 6.29115e-11
Epoch: 99/100 Loss: 6.29115e-11
Epoch: 100/100 Loss: 6.29115e-11
```

[8]:
```
import matplotlib.pyplot as plt
y = range(0,100)
plt.plot(y, sgd_loss, label = "SGD")
plt.plot(y, adam_loss, label = "ADAM")
plt.plot(y, lbfgs_loss, label = "L-BFGS")
plt.legend()
plt.show()
```



[ ]: