test_functions

April 24, 2023

[ ]:

```
import torch
import numpy as np
import matplotlib.pyplot as plt
from torch import nn
from torch.autograd import Variable
import copy
```

[ ]:

```
d = 100

def synthetic_example(iters=100_000, lr=1e-3):
    # Objective function
    def func(x):
        val = 0
        for i in np.arange(d - 1):
            val += (100*(x[i + 1] - x[i]**2)**2 + (x[i] - 1)**2)
        return val

    x0 = np.random.uniform(-2.048, 2.048, d)

    x_Adam = Variable(torch.tensor(x0), requires_grad=True)
    x_AMS  = Variable(torch.tensor(x0), requires_grad=True)
    x_SGD  = Variable(torch.tensor(x0), requires_grad=True)

    # avg_regret_checkpoints = []
    # iteration_checkpoints  = []
    # x_checkpoints          = []

    optimizer_Adam = torch.optim.Adam([x_Adam], lr=lr, betas=(0.9, 0.999),␣
 ↪eps=1e-08, amsgrad=False)
    optimizer_AMSGrad = torch.optim.Adam([x_AMS], lr=lr, betas=(0.9, 0.999),␣
 ↪eps=1e-08, amsgrad=True)
    optimizer_SGD  = torch.optim.SGD([x_SGD], lr=lr, momentum=0.9)

    # total_regret = 0

    for iter in np.arange(1, iters + 1):
        loss_Adam     = func(x_Adam)
        loss_AMS      = func(x_AMS)
        loss_SGD      = func(x_SGD)

        # total_regret += np.linalg.norm(loss.item() - x_true)

        # if (iter % 10000 == 0):
        #     avg_regret = total_regret / iter
        #     avg_regret_checkpoints.append(avg_regret)
        #     iteration_checkpoints.append(iter)
        #     x_checkpoints.append(x.item())

        optimizer_Adam.zero_grad()
        loss_Adam.backward()
        optimizer_Adam.step()

        optimizer_AMSGrad.zero_grad()
        loss_AMS.backward()
        optimizer_AMSGrad.step()

        optimizer_SGD.zero_grad()
        loss_SGD.backward()
        optimizer_SGD.step()

    return x_Adam, x_AMS, x_SGD
```

```
[ ]: x_true = torch.tensor(np.ones(d))

     iters = 100000
     lr    = 1e-4
     x_Adam, x_AMS, x_SGD = synthetic_example(iters=iters, lr=lr)

     print(f"2-norm between Adam x and true x:    {torch.linalg.vector_norm(x_Adam -␣
       ↪x_true)}")
     print(f"2-norm between AMSGrad x and true x: {torch.linalg.vector_norm(x_AMS -␣
       ↪x_true)}")
     print(f"2-norm between SGD x and true x:     {torch.linalg.vector_norm(x_SGD -␣
       ↪x_true)}")
```

```
2-norm between Adam x and true x:    1.9032832877996656e-05
2-norm between AMSGrad x and true x: 8.296030842953964
2-norm between SGD x and true x:     1.4430469435194013e-13
```

```
[ ]: print(f"Infinity-norm between Adam x and true x:    {torch.linalg.
       ↪vector_norm(x_Adam - x_true, float('inf'))}")
     print(f"Infinity-norm between AMSGrad x and true x: {torch.linalg.
       ↪vector_norm(x_AMS - x_true, float('inf'))}")
```

```
print(f"Infinity-norm between SGD x and true x:     {torch.linalg.
 ↪vector_norm(x_SGD - x_true, float('inf'))}")
```

```
Infinity-norm between Adam x and true x:    5.129391164926389e-06
Infinity-norm between AMSGrad x and true x: 1.7579907796582186
Infinity-norm between SGD x and true x:     1.2501111257279263e-13
```