ML4Science:
Week 4 Meeting

Calafà, M., Mescolini, G., Motta, P.

École Polytechnique Fédérale de Lausanne (EPFL)

Machine Learning Project 2

14 Dec 2021

Tutor: Dr. Michele Bianco

Outlines

Lightening the computational effort (part 2)

Data cleaning

With the garbage collector (`gc.collect()`) we cleaned run-time the memory previously allocated for:

- Macro-tensor X (with all data from the dataset)
- Sub-tensors of X (train, test ...)
- Temporary datasets

In order to keep in memory the data-loader only. Many GB saved, lower risk of sudden kernel stops.

Process reloading

Process reloading

The training process is very slow if repeated for many epochs.

Idea: save locally the main parameters at the end of every epoch.

Two benefits:

1. Possibility to stop the computation and restart it later
2. Backup of the information for undesirable breaks

Process reloading

It is important to distinguish between the very first run from the following reloadings:

```

if (first_run == True):
        net = CNN.CNN()
        optimizer = optim.Adam(net.parameters(), lr=1e-3)
        current_epoch = 0
        final_epoch = epochs
        prev_loss = 10**2 # high initial value
        all_test_losses = []
        all_train_losses = []
        all_R2_train = []
        all_R2_test = []
    else:
        #Resume the training
        PATH = '.\model\last_model.pt'
        net = CNN.CNN()
        optimizer = optim.Adam(net.parameters(), lr=1e-3)
        checkpoint = torch.load(PATH)
        net.load_state_dict(checkpoint['model_state'])
        optimizer.load_state_dict(checkpoint['optimizer_state'])
        scheduler.load_state_dict(checkpoint['scheduler_state'])
        current_epoch = checkpoint['epoch'] + 1
        prev_loss = checkpoint['loss']
        final_epoch = current_epoch + epochs
        ...
```

Other additions

Plots

We added the correlation plot ($xi_{predicted}$ vs xi_{true}) and the loss trend with respect to the epoch, both for training and for testing.

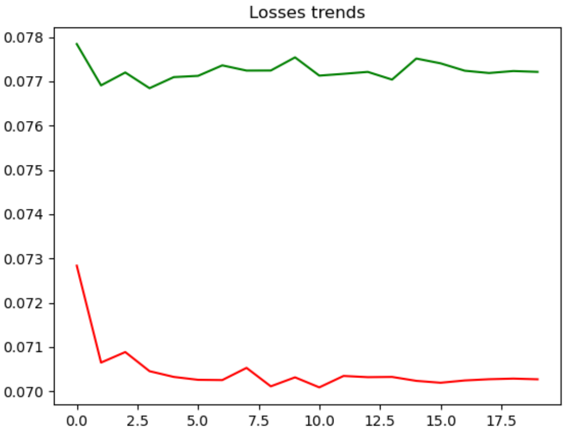
```

def correlation_plot(x_pred, x_true):
    plt.plot(x_true, x_true, 'r') # y = x
    plt.plot(x_true, x_pred, 'b') # our actual prediction
    sigma = np.std(x_pred)
    plt.plot(x_true, x_pred + sigma, 'r-')
    plt.plot(x_true, x_pred - sigma, 'r-')

def plot_losses(epochs, loss_tr, loss_te):
    plt.plot(epochs, loss_tr, 'r') # training losses
    plt.plot(epochs, loss_te, 'g') # test losses
    plt.title('Losses trends')
    plt.show()
```

Plots

Here we report the above mentioned loss plot obtained with our results (green is test, red is training):



Other additions

Scheduler

```
from torch.optim.lr_scheduler import ReduceLROnPlateau
# ...
scheduler = ReduceLROnPlateau(optimizer = optimizer, mode = 'min', factor
    = 0.1, patience = 7, min_lr = 1e-7)
# ...
scheduler.step(loss_test)
```

R2 score

```
from sklearn.metrics import r2_score
# ...
R2 = r2_score(y_train.detach(),output.detach())
```

Extraction of Validation set

```
# split dataset into training (2500), validation set (500) and prediction
    set (300)
X_train, X_pred, y_train, y_pred = train_test_split(X, y, test_size=0.1,
    random_state=2021)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
    test_size=0.2, random_state=2021)
```

Issues and Questions

Issues

1. Issues with Paolo's Laptop: we could only reach 20 epochs. Is there a possibility of usage of EPFL Cluster?
2. About February conference