CSC3002S Capstone Project

README: Hyperparameter Optimisation

Group:

1. RSNJOS005 --- Joshua Rosenthal
2. FSHJAR002 --- Jarred Fisher
3. WLSKIE001 --- Kiera Wilson

Instructions for Dockerfile build and running our project:

Our program does not have a GUI, as our system is not orientated towards constant user interaction. We have however made it as user friendly as possible to run our program by implementing a script file that handles the setup of all neccessary docker components and gives the user an easy entrypoint to specify a problemcard and timeout.

If running natively from **Linux** based system:
(Recommended simply because the bash script is more versatile than the windows one)

Run the bash script autorun.sh. This can easily be done by typing the following in the project directory:

```
./autorun.sh [options]
```

An example run would be:

```
./autorun.sh -z test1.wcard -t 2
```

This would run our project with problem card = test1.wcard and the timeout = 2 (seconds)
Note: The order of flags are not important. Furthermore, only supply the problem card name plus the .wcard extension as seen in the above example. File paths are handled internally.

To see a list of all options available call the script as follows:

```
./autorun.sh -h
```

This will display the following help message:

```
USAGE:
./autorun.sh [options]
OPTIONS:
-z : ARG Problem wcard, no default value. Must be specified if -i flag not passed. Just give t
-t : ARG Timeout for CarlSAT, default = 2. Should be specified along with -z flag.
-h              Display this help message
-i              Enable run container interactively, default = false. Must be specified if not
-u              Update local hyperopt database with gitLab synched sqlscripts/hyperopt.sql DB
-f              Remove associated docker image and rebuild from scratch. i.e. docker image is
```

```
NOTE:
The -i flag is mutually exclusive from the -z and -t flags. The -i flag will not directly cal
as interactive mode is intended for development and debugging. If -i is not passed, the docke
and call the Wrapper class directly with relevant output being printed to host console. The -
within the Docker container.
The flags -u and -f can be combined with both interactive and non-interactive mode.

EXAMPLES:
./autorun.sh -z test1.wcard -t 2 -f -u
./autorun.sh -i -u
```

If running natively from **Windows** system:

> Run the batch script windows_autorun.bat. Similarly, this can be done by typing the following in the project directory:
>
> ```
> .\windows_autorun.bat problemcard timeout
> ```
>
> **Note**: The windows batch script does *not* take in flags like the linux bash script does. The windows based script file expects a problem card and timeout value in the order as shown above. This is due to batch scripts not having flag like functionality such as getopts (linux).
>
> An example run would be:
>
> ```
> .\windows_autorun.bat test1.wcard 2
> ```
>
> This would run our project with problem card = test1.wcard and the timeout = 2 (seconds)
> **Note**: Only supply the problem card name and .wcard extension as seen in the above example. File paths are handled internally.

Developer notes:

Quick reference (usual order of operation):

1. Pull from git
2. (*OPTIONAL*) pass -u flag to ./autorun.sh (if you want to sync your local hyperopt DB with the git repo one).
3. Run `./autorun.sh -z problemcard -t timeout`
4. When work is complete and you want to stage files and commit changes. Run ./gitAuto.sh then write commit message, then push.

Further information is below (mainly for the developers):

Notes on script files

Running autorun.sh does the neccessary building of an image, creating a volume (for persisting data), starting the containers and then runs the container. It does some relevant checks to see if these docker components already exist and adjusts accordingly (for example stopping and removing the container if it already exists before booting it back up).

The windows batch script file does not have all the nice functionality as the linux based one has. It is there for

completeness and portability.

The gitAuto.sh script stages all relevant files in the project directory and it also creates an sql dump (basically copies your DB) and puts it in two directories - where both form part of the git repo. The first is into sqlscripts/hyperopt.sql. This overwrites that file with your local hyperopt DB, meaning that the next time someone pulls from the repo and updates their local hyperoptDB, they will be using your most recent version of the hyperopt DB. This is how we create a semblance of a concurrent database.

Furthermore, the gitAuto script makes a backup of your local hyperopt DB and adds it to the dbBackup directory. Every backup is unique, and is date and time stamped.

The linux autorun.sh script will search for a docker image by the name of "dimage" and if it does not find it, it will build the image from the dockerfile always regardless of parameters passed to the script.

By default, even if an image is present it'll still be rebuilt. This is because any changes made to the source files would not reflect in the docker environment if the image wasn't rebuilt.

When you pull from the git it will not overwrite any data/ db entries you currently have stored on your docker volume) However, if you want to update your local version of the hyperopt database, pass -u flag to ./autorun.sh. This will overwrite the hyperopt database that you currently have stored on your local docker volume with the latest one from the repo. It's basically restoring your DB from an sql dump. This IS NOT a merging of the databases. It is completely overwriting what you had previously with what is in sqlscripts/hyperopt.sql