4. Code organization

The code for the neural networks SDF and beta estimation exercise is under src/NN. The code with example files for the preprocessing is under `src/NN/PreprocessingFiles`. Finally, `src/NN/SlurmScripts` contains slurm scripts for running on a cluster.

The code for the GMM exercise is under `src/GMM`.

The code for the Gaussian HMM is under `src/HMM` (one file only).

See the file requirements.txt for the packages under which code has been tested. Need to change paths as needed to run the scripts.

4.1 How to run

First create a conda environment with minimal requirements as in `requirements.txt`.

The slurm submission files for neural networks estimations of the SDF are similar, except for the change in the training script called. They are contained in `src/SlurmScripts`. Change the file `activate_env.sh` under `src/NN` as needed. It is cluster-specific and loads the environment in each slurm script contained in `src/NN/SlurmScripts`.

Note: the beta net estimation allows in addition for data parallelism across multiple GPUs on one cluster node. This has been tested. We do not post a version of the SR-max, MP-min or GAN networks with data parallelism since they have not yet been tested.

- For the SDF estimation using SR-max, MP-min: First run a respective preprocessing file producing the inputs for the neural networks exercise. Then, for the SDF estimation, change the job numbers and the project_name accordingly in the `slurm_sdf_estimation.sh` under `src/SlurmScripts`. Then submit `slurm_sdf_estimation.sh` with the sbatch command:

  ```
  sbatch slurm_sdf_estimation.sh
  ```

  This slurm script uses the train script `nolabels_one_shot.py`. In one fell swoop, it will run the whole hyperparameter and architecture loop, as well as find best model+hyperparameters and produce and save ensembling, together with its performance.

- For the beta estimation, copy first the respective file of beta labels produced by the SDF estimation into the data folder used by the preprocessing file of the beta estimation. Then run the preprocessing file. Finally submit the slurm job via

  ```
  sbatch slurm_beta_estimation.sh
  ```

  with any necessary changes to project_name, job-number, save paths etc. Just as for the SDF estimation, this will run the grid search, as well as find best hyperparameters+architecture, and do the ensembling in one fell swoop. This uses the train script `labels_one_shot.py` that allows for data parallelism on one multi-gpu node.

- For the GAN network procedure: similar to case above, except now you submit the slurm file

  ```
  sbatch slurm_gan_estimation.sh
  ```

  This uses the train script `one_shot_gan.py`.

- For the GMM exercise: run first `GMM_bondpfs_preprocessing.py`. Determine there the number of PCA factors one wants to run by running `bai_ng_analysis()`. Then change the respective `nr_facs` variable in `preprocess_ejnports` function to save the respective input dataframes for the GMM exercise. Finally, run `GMM_bondpfs_main.py`. For the Diebold-Mariano test analysis of residuals between neural networks and predictive regressions, run `GMM_vs_NN_residuals.py`. ## 4. Code organization {#codeorga}

The code for the neural networks SDF and beta estimation exercise is under src/NN. The code with example files for the preprocessing is under `src/NN/PreprocessingFiles`. Finally, `src/NN/SlurmScripts` contains slurm scripts for running on a cluster.

The code for the GMM exercise is under `src/GMM`.

The code for the Gaussian HMM is under `src/HMM` (one file only).

See the file requirements.txt for the packages under which code has been tested. Need to change paths as needed to run the scripts.

4.1 How to run

First create a conda environment with minimal requirements as in `requirements.txt`.

The slurm submission files for neural networks estimations of the SDF are similar, except for the change in the training script called. They are contained in `src/SlurmScripts`. Change the file `activate_env.sh` under `src/NN` as needed. It is cluster-specific and loads the environment in each slurm script contained in `src/NN/SlurmScripts`.

Note: the beta net estimation allows in addition for data parallelism across multiple GPUs on one cluster node. This has been tested. We do not post a version of the SR-max, MP-min or GAN networks with data parallelism since they have not yet been tested.

- For the SDF estimation using SR-max, MP-min: First run a respective preprocessing file producing the inputs for the neural networks exercise. Then, for the SDF estimation, change the job numbers and the project_name accordingly in the `slurm_sdf_estimation.sh` under `src/SlurmScripts`. Then submit `slurm_sdf_estimation.sh` with the sbatch command:

```
sbatch slurm_sdf_estimation.sh
```

This slurm script uses the train script `nolabels_one_shot.py`. In one fell swoop, it will run the whole hyperparameter and architecture loop, as well as find best model+hyperparameters and produce and save ensembling, together with its performance.

- For the beta estimation, copy first the respective file of beta labels produced by the SDF estimation into the data folder used by the preprocessing file of the beta estimation. Then run the preprocessing file. Finally submit the slurm job via

  ```
  sbatch slurm_beta_estimation.sh
  ```

  with any necessary changes to project_name, job-number, save paths etc. Just as for the SDF estimation, this will run the grid search, as well as find best hyperparameters+architecture, and do the ensembling in one fell swoop. This uses the train script `labels_one_shot.py` that allows for data parallelism on one multi-gpu node.

- For the GAN network procedure: similar to case above, except now you submit the slurm file

  ```
  sbatch slurm_gan_estimation.sh
  ```

  This uses the train script `one_shot_gan.py`.

- For the GMM exercise: run first `GMM_bondpfs_preprocessing.py`. Determine there the number of PCA factors one wants to run by running `bai_ng_analysis()`. Then change the respective `nr_facs` variable in `preprocess_ejnports` function to save the respective input dataframes for the GMM exercise. Finally, run `GMM_bondpfs_main.py`. For the Diebold-Mariano test analysis of residuals between neural networks and predictive regressions, run `GMM_vs_NN_residuals.py`.