README — version 2
Data repository for the paper
"Binary Classification as a Phase Separation Process"

Rafael Monteiro MONTEIRODASILVA-RAFAEL@AIST.GO.JP,
RAFAEL.A.MONTEIRO.MATH@GMAIL.COM

Mathematics for Advanced Materials Open Innovation Laboratory,
AIST, c/o Advanced Institute for Materials Research,
Tohoku University, Sendai, Japan

Abstract

This is a README file with general instructions for users of the data repository Monteiro (2021a), containing the code and the data used in the paper "Binary Classification as a Phase Separation Process", by Rafael Monteiro.

This guide explains what is stored in the data repository, how computational statistics can be retrieved, and which files to use to replicate the data we did statistics on and generated with our code. We explain "what" is stored, mostly refraining from explaining "how" such a data is generated: for that, the reader can find a thorough explanation of that either in the paper or in the implemented code (or better, reading both).

All the data we refer to is available at Monteiro (2021a),

https://dx.doi.org/10.5281/zenodo.4005131.

All the code we refer to is available at Monteiro (2021b),

https://github.com/rafael-a-monteiro-math/Binary_classification_phase_separation.

For matters of reproducibility, the code does not require proprietary software: you simply need Python, which is free and available online.

Contents

Part I
Preliminaries

1. How to open the data

We will assume that you have some basic knowledge: how to run some python code on a jupyter-notebook and how to use the shell in an UNIX based system.

1.1 Decompressing tarballs

All the data are inside folders as compressed tarballs, with extension ".tar.gz". For this you will need the dependency GNU tar.

For example, if **example.tar.gz** is the name of the file you want to decompress, you run the following command on your terminal[1]

```
tar -xzvf example.tar.gz
```

with a similar command for the other cases.

1.2 Reading pickled files

All the data has been recorded in dictionaries, statically stored as ".p" files. You can access these files using python in the following manner: first, you need to open python and import the following package

```
try:
    import cPickle as pickle
except ImportError: # if you use python 3.x
    import pickle
```

Then, assuming that you want to read a file **data.p** in the folder **Statistics/MNIST/**, you run

```
with open("Statistics/MNIST/parameters_MNIST_Neumann.p", "rb") as fp:
      import_dictionary = pickle.load(fp)
```

and now you have the dictionary **import_dictionary** available to you. More examples of dictionaries usage can be found in the git-hub repository for this paper "Monteiro (2021b)".

1.3 Main python dependencies

The whole code was run using Python 3.5.2 and Python 3.6. Modules version are given below:

Versions used were as follows:

- Numpy 1.18.5
- Tensorflow 2.2.0
- Keras 2.3.0-tf
- Sklearn 0.20.3
- Scipy 1.4.1
- jupyter-notebook : 6.4.0

1. I'm assuming an UNIX based system, like Linux or MAC.

Part II

Content in the folder PSBC

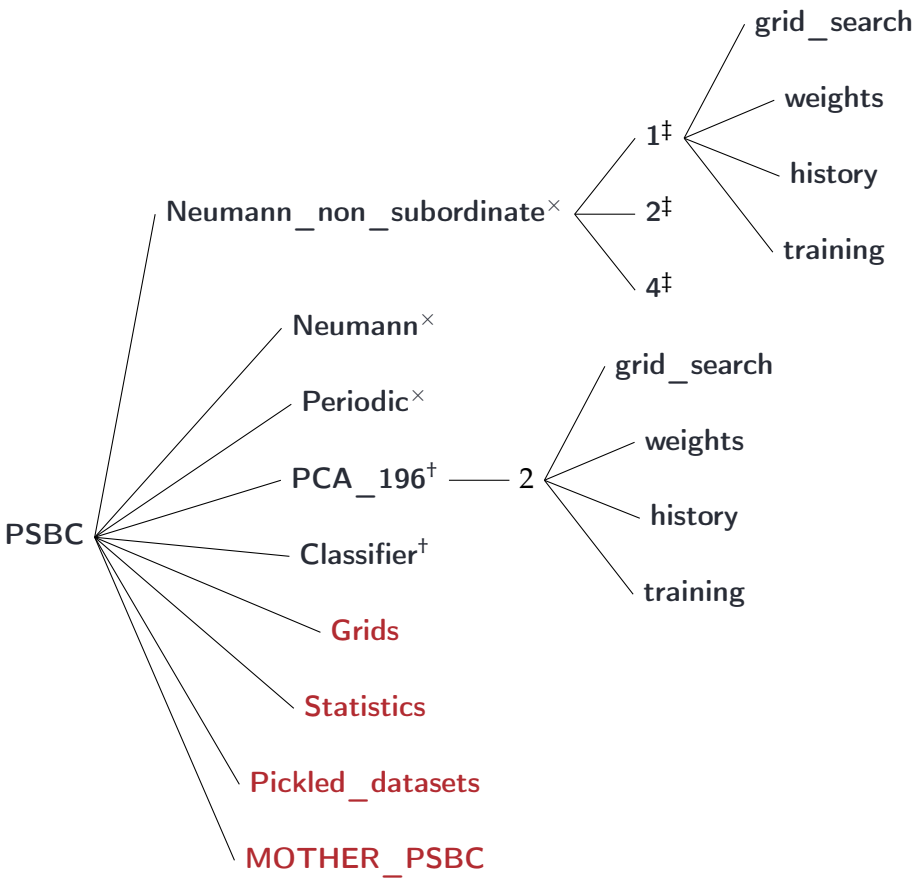

Figure 1: Contents in the folder **PSBC**. To make the tree less poluted, we use a convention where folders with the same upper mark have the same subfolder structure. The subfolder **grid_search** of the folder **Periodic** is a copy of that in **Neumann** (because of the way model selection takes place; see the paper, Section 3).

In figure 1 we have the full folder structure for the PSBC tests. There are still a few other files inside the **PSBC** folder: some of them are jupyter-notebooks, explained later in Part III; others are python scripts, described below.

- **generate_parameters_for_grid_search.py** : this file creates the folder **Grids**, where you will find several pickled files. This file is responsible for writing these parameters according to some variables: grid search or training, Neumann boundary conditions (as True or False), and the number of Layers used. The files are them pickled and put inside the folder **Grids**, whose content we will explain later.
- **create_grid_for_indexes.py** : this file creates the a pickled file enumerating pairs of distinct digits in the set $\{0,\ldots,9\}$. The output is the file **digits_index.p**, saved in the folder **Grids** (see Section 2.4).
- **mnist_pickled.py** : this is a program with an argument. It can be called as

```
python 3.* mnist_pickled.py XX
```

where XX is either MNIST or any different string.

This program splits the MNIST data set in train and test set, then do a further partition according to labels. Each of these partition will be pickled and saved. For instance, the file **X_train_0.p** contains the subset of elements with label 0 in the training set.

Since we do kfold cross validation in many subsets of the MNIST dataset, we separate the indexes in each kfold in advance, that is, in each kfold, which elements are going to be used for optimization and which will be used for testing. In that way, tests comparing different architectures (in the case, Subordinate and Non-subrdinate, and also PCA versus Non-PCA) will go through grid search on the same elements, and variations between different tests will not depend on the set chosen.

The outputs of this program are saved in the folder **Pickled_datasets** (see Section 2.6).

- **PSBC_statistics.py** : script that summarizes and generate statistics from the test results. All the outputs are put in the folder **Statistics**.

It is easy to math the each tree branch to the corresponding model. For instance, if one follows the branch

$$\textbf{PSBC} > \textbf{Neumann\_non\_subordinate} > \textbf{1} > \textbf{weights} \tag{1}$$

one finds the data corresponding to the PSBC model with Neumann boundary conditions and non-subordinate architecture, with 2 layers. The roles of **1** and **weights** will be clarified in Section 2.2.

2. How to read this tree and the content in its leaves

The tree was made less poluted by using the convention where folders with the same upper mark have the same subfolder structure. For instance, both folders **Neumann_non_subordinate** and **Periodic** have subfolders with same names and structures. Hence, since a path like (1) exists, one also finds a path

$$\textbf{PSBC} > \textbf{Periodic} > \textbf{1} > \textbf{weights}\ . \tag{2}$$

Using this convention, we need to explain just a feel cases. Luckily, regardless of the branch one is in, folders with the same name have a same nature, but are associated with a different PSBC architecture. We summarize their properties next.

2.1 Content in the folders 1, 2, 4

The name of these folders indicate the number of layers N_t in the PSBC model they are storing data from.

2.2 Content in the folders history, grid_search, weights, and training

- Each folder **grid_search** contains the output of grid searches as a pickled file. For example, in the folder

  Classifier_196/2/grid_search/

  one will find files with name like

  Normal_all_grid_search_results7_9.p,

  which contains the results of the grid search for hyperparameters for the dataset with labels 7 and 9. The name of this file changes slightly in tests where PCA was used.

- Each folder **weights** contains ".h5" files, where weights to models are stored, and pickled dictionaries ".p", where accuracies at the validation data (in the case, the test set) are stored. For example, one will find files like

  0_0_49_1_0_0_fold_0_Index_1_49_1_best_weights.h5,

  and

  0_0_49_1_0_0_fold_0_Index_1_49_1_val_accuracy.p.

We won't go into details of what each one of these numbers mean, but they are used to indicate which hyperparameters and which of the 5 folds during model selection (or 5 replicas of experiments, in case of model assessment) are stored there.

For technical details, we refer the reader to the code in Monteiro (2021b), in particular the function

my_gridSearch_with_index

in the file

tf_PSBC_extra_libs_for_training_and_grid_search.py,

which is in the folder **MOTHER_PSBC** (see Section 2.3).

- Each folder **history** contains several pickled dictionaries with uninviting names, like

  0_0_196_1_0_0_fold_0_PCA__var0_3_var1_5.p

  and similar things. Each one of these dictionaries contain information obtained during training, like the evolution of Δ_t^u and Δ_t^p over epochs, the evolution of $\max_{0\leq k\leq N_t-1}\left(\|W_u^{\lfloor k\rfloor}\|_{\ell^\infty}\right)$ and $\max_{0\leq k\leq N_t-1}\left(\|W_p^{\lfloor k\rfloor}\|_{\ell^\infty}\right)$, the loss function at the end of each epoch, and the classifier accuracy after prediction on the designated validation set passed to the function **my_gridSearch_with_index**, also mentioned above.

- Each folder **training** has two types of pickled dictionaries, corresponding to model assessment step. Data collected using the training set have names like

  Training_accuracies_1_98_1_vary_eps_0_1.p,

  while data from the test set have names like

  Test_set_accuracies_1_49_1_vary_eps_0_1.p.

  The structure of both dictionaries is similar: in the first entry there is a a tensor (a multi-dimensional matrix) $\mathbb{M}$ with the accuracies of each experiment. In the second entry there is another dictionary where, roughly speaking, each key indicates the the hyperparameters associated with the accuracies stored in in $\mathbb{M}$.

  In this second dictionary one also finds a "dummy" key $(-1,-1,-1,-1,-1,-1)$, explaining the order of parameters stored in $\mathbb{M}$: **('eps', 'dt', 'pt_cardnlty', 'layers_K_shared', 'lr_U', 'lr_P'),**, each of them indicating how the variables (ε, Δ_t^p (and Δ_t^u), N_{pt}, layers-k-shared, learning rate for U equation, learning rate for P equation, are initialized.

  Last, a small correction: if you note, $\mathbb{M}$ has 7 entries, but there are only 6 stored parameters. The reason is that the first entry in $\mathbb{M}$ concerns the 5 folds (resp., replicas) during model selection (resp., model assessment); the last 6 entries are as described above.

2.3 Content in the folder MOTHER_PSBC

This folder contains all the libraries used to run the model. It also contains python scripts and bash scripts, as detailed below.

- **tf_PSBC_extra_libs_for_classifier.py** is a python script with extra functions and models created for the multiclass classification. The model created using Ensemble learning is implemented here.

  We remark that this file relies on the folder structure shown in Tree 1, because it accesses trained weights.

- **tf_PSBC_extra_libs_for_training_and_grid_search.py** is a python script with extra functions used in training amd grid search.

- **tfversion_binary_phase_separation.py** is the main library of this PSBC implementation, where all the layers, are built, concatenated into smaller structures, and put together into the PSBC model.

- **PSBC_plots_useful.py** contains some useful function used for plotting figures.

2.4 Content in Grids

In the folder MNIST you will find many pickled files. The first two that we discuss are files of the type

A_B_C.p,

where **A** $\in$ {**grid_search**, **training**}, **B** $\in$ {**True**, **False**}, and **C** $\in \{1, 2, 4\}$.

2.5 Content in Statistics

This folder contains pickled files with all the simulations or summaries of them. They are all outputs of the script **PSBC_statistics.py**.

Since most of them have the same structure, we will use some conventions: we shall say that key_1 is a key to the dictionary $\mathcal{A}$ in its 1th-key-layer if $\mathcal{A}[\text{key}_1]$ is valid. In the general case, we shall say that a key key_k of a dictionary $\mathcal{A}$ is in its kth-key-layer if, given keys key_j in its first $k-1$ layers, the operation

$$\mathcal{A}[\text{key}_1][\text{key}_2]\dots[\text{key}_{k-1}][\text{key}_k] \tag{3}$$

is valid.

Now we are ready to explain the dictionaries in this folder. For a hands-on usage and manipulation of these objects, see the notebook "(Monteiro, 2021b, **PSBC_using_statistical_files.ipynb**)".

- **All_accuracies.p** Contains the accuracies to all carried out simulations. Its layers are the following
  - 1-key-layer concerns folder names. They are

    $$\text{key}_1 \in \{\texttt{"Neumann"}, \texttt{"Periodic"}, \texttt{"Neumann\_non\_subordinate"}, \texttt{"Classifier\_196"}, \texttt{"PCA\_196"}\}. \tag{4}$$

  - 2-key-layer concerns number of layers N_t. They are

    $$\text{key}_2 \in \begin{cases} \{2\} & \text{if} \quad \text{key}_1 \in \{\texttt{"Classifier\_196"}, \texttt{"PCA\_196"}\}, \\ \{1,2,4\} & \text{otherwise.} \end{cases} \tag{5}$$

  - 3-key-layer concerns the pair of digits considered. They vary according to the previous layer:

    $$\text{key}_3 \in \begin{cases} \{(0,1),\dots,(8,9)\} & \text{if} \quad \text{key}_1 = \texttt{"Classifier\_196"} \\ \{(3,5),(4,9)\} & \text{if} \quad \text{key}_1 = \texttt{"PCA\_196"}, \\ \{(0,1)\} & \text{otherwise.} \end{cases} \tag{6}$$

    The key key_3 obeys a constraint: pairs (a,b) are always in the form $a < b$, so as to be uniquely represented. Obviously, $a \neq b$, since things like $\text{key}_3 = (1,1)$ are meaningless.
  - 4-key-layer concerns layers-k-shared. They vary as

    $$\text{key}_4 \in \begin{cases} \{1\} & \text{if} \quad \text{key}_1 \in \{\texttt{"Classifier\_196"}, \texttt{"PCA\_196"}\}, \\ \{(1, N_t\} & \text{otherwise, where} \quad N_t = \text{key}_2. \end{cases} \tag{7}$$

  - 5-key-layer the type of data.

    $$\text{key}_5 \in \{\texttt{'train\_set\_std'}, \texttt{'train\_set'}, \texttt{'test\_set'}, \texttt{'test\_set\_std'}, \texttt{'test\_set\_mean'}, \texttt{'train\_set\_mean'}\}. \tag{8}$$

When used as in (3), we obtain the accuracies associated to a PSBC with associated architecture. For instance,

$$\text{All\_accuracies}[\texttt{"Neumann"}][2][\texttt{(0,1)}][2][\texttt{'train\_set'}]$$

contains all the training set accuracies in the 5 model assessments performed for PSBC with Neumann Boundary conditions, $N_t = 2$, $layers - 2 - shared$, while

$$\text{All\_accuracies}[\texttt{"Neumann"}][2][\texttt{(0,1)}][2][\texttt{'train\_set\_mean'}]$$

gives the mean of such results. Obviously, (0,1) concerns to the digits used in these tests.

- **Best_parameters.p**
  This dictionary contains the values of best hyperparameters for the grid search we carried out. Its first three keys are as in (4), (5), (6), and (7).
  * Its 5th is
  $$\text{key}_5 \in \begin{cases} \{196\}, & \text{if} \quad \text{key}_1 \in \{\texttt{"Classifier\_196"}, \texttt{"PCA\_196"}\}, \\ \left\{\lfloor \frac{784}{k} \rfloor, \text{ for } 1 \leq k \leq 5\right\}, & \text{otherwise.} \end{cases} \tag{9}$$
  * Its 6th is
  $$\text{key}_6 \in \left\{ \begin{array}{c} \texttt{'lr\_U\_range'}, \texttt{'eps\_range'}, \texttt{'lr\_P\_range'}, \\ \texttt{'layer\_share\_range'}, \texttt{'ptt\_range'}, \texttt{'dt\_range'} \end{array} \right\} \tag{10}$$
  When used as in (3), we obtain the hyperparameters of a PSBC with associated architecture. For instance,
  $$\text{Best\_parameters}[\,\texttt{"Neumann"}][\,\texttt{2}][\,\texttt{(0,1)}][\,\texttt{2}][784]$$
  contains

```
'dt_range': array([0.2], dtype=float32),
'eps_range': array([0.    , 0.0625, 0.125 , 0.25  , 0.5   , 1.    ],␣
↪dtype=float32),
'layer_share_range': array([2], dtype=uint16),
'lr_P_range': array([0.1], dtype=float32),
'lr_U_range': array([0.001], dtype=float32),
'ptt_range': array([784], dtype=uint16)
```

  With values for Δ_t^u and Δ_t^p in **'dt_range'** (recall that they are initialized with the same value), ε, to be chosen from **'eps_range'**, layers-k-shared to be chosen from learning rate for P system to be chosen from **'lr_P_range'**, learning rate for U system to be chosen from **'lr_U_range'**, N_{pt} to be chosen from **ptt_range'**.
- **All_filepaths_best_weights.p** This dictionary contains the path to the weights of each PSBC model that has been trained. Its first three keys are as in (4), (5), and (6), its 4th keys is as in (9), and 5th keys is as in (9).
  * It's 6th key concerns values of the parameter ε. When it takes the value **'info'** it displays information about the other values this key takes, which are
  $$\text{key}_6 \in \begin{cases} \{0\}, & \text{if} \quad \text{key}_1 \in \{\texttt{"Classifier\_196"}, \texttt{"PCA\_196"}\}, \\ \{0, \ldots, 5\}, & \text{otherwise.} \end{cases} \tag{11}$$
  with the convention that $\text{key}_5 = 0$ corresponds to $\varepsilon = 0$, and to $\varepsilon = 2^{(\text{key}_5 - 5)}$ for other values.
  * A 7th key then follows,
  $$\text{key}_7 \in \{\texttt{'best\_weights'}, \texttt{'val\_acc'}\}. \tag{12}$$
  For instance, the paths in the tree 1 for the 5 realization of the PSBC with Neumann boundary conditions, $N_t = 1$, layers-1-shared, $N_{pt} = 784$, $\varepsilon = \frac{1}{2^3}$ can be accessed at
  $$\text{All\_filepaths\_best\_weights}[\,\texttt{"Neumann"}][\,\texttt{1}][\,\texttt{(0,1)}][\,\texttt{1}][784][\,\texttt{2}][\,\texttt{'best\_weights'}]$$
  . The stored content is

```
['Neumann/1/weights/2_0_784_1_0_0_fold_0_Index_1_784_1_best_weights.h5',
'Neumann/1/weights/2_0_784_1_0_0_fold_1_Index_1_784_1_best_weights.h5',
'Neumann/1/weights/2_0_784_1_0_0_fold_2_Index_1_784_1_best_weights.h5',
'Neumann/1/weights/2_0_784_1_0_0_fold_3_Index_1_784_1_best_weights.h5',
'Neumann/1/weights/2_0_784_1_0_0_fold_4_Index_1_784_1_best_weights.h5']
```

- **All_histories.p**
  This dictionary contains the training history for each PSBC we fitted during model assessment. Its first 6 keys in are as those in **All_filepaths_best_weights.p**. For instance,

$$\text{All\_histories}[\,\text{"Classifier\_196"}][\,2][\,(8,9)][\,1][\,196][\,0] \tag{13}$$

  is valid, corresponding to a PSBC model with architecture $N_t = 2$, layers-1-shared, $N_{pt} = 196$, $\varepsilon = 0$, associated to the digits 8 and 9.

  * A 7th key then is introduced:

$$\text{key}_7 \in \{0, \ldots, 4\}\,. \tag{14}$$

    pointing out to one of the 5 model assessments we carried out. For instance the output of[2]

$$\text{All\_histories}[\,\text{"Classifier\_196"}][\,2][\,(8,9)][\,1][\,196][\,0][0] \tag{15}$$

    is (truncated)

```
'dt_P': [0.005502431187778711,
0.003485201857984066,
...
0.0023186421021819115],
'dt_U': [0.5699999928474426,
0.5699999928474426,
...
0.5699999928474426],
'w_p_inf': [10.177945137023926,
12.788613319396973,
...
15.667551040649414,
15.679079055786133],
'w_u_inf': [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

- **Snapshot_best_weights.p**
  This dictionary contains the weights of the PSBC model represented as W_u and W_p in the paper. Its first 6 keys in are as those in **All_histories.p**.

  * Its 7th key is

$$\text{key}_7 \in \{\,\text{'P'},\ \text{'U'}\}\,. \tag{16}$$

    For instance, objects like

$$\text{Snapshot\_best\_weights}[\,\text{“Neumann"}][\,2][\,(0,1)][\,1][\,49][\,0][\,\text{"P"}]$$

    are lists of five elements, retrieved as in an array. Thus, for example,

$$\text{Snapshot\_best\_weights}[\,\text{“Neumann"}][\,2][\,(0,1)][\,1][\,49][\,0][\,\text{"P"}][0]$$

    is a matrix.

- **Snapshot_best_weights_stats.p** It's first 4 layers are as in (4), (5), (6), (7).

  * Its 5th key is

$$\text{key}_5 \in \left\{ \begin{array}{c} \text{'diameter\_P', 'mean\_min\_U', 'min\_P', 'mean\_max\_U', 'std\_max\_P',} \\ \text{'mean\_max\_P', 'mean\_min\_P', 'std\_min\_P', 'diameter\_U', 'max\_P',} \\ \text{'std\_diameter\_P', 'min\_U', 'std\_diameter\_U', 'std\_max\_U',} \\ \text{'mean\_diameter\_U', 'mean\_diameter\_P', 'std\_min\_U', 'max\_U'} \end{array} \right\} \tag{17}$$

2. The term in red has been used to distinguish (15) from (13).

There are many key values. For details, see "(Monteiro, 2021b, PSBC_statistics.py)", in particular the functions

create_snapshot_best_weights () and **create_snapshot_best_weights_stats ()**.

For instance, when $\text{key}_1 \in \{$ **"Classifier_196"**, **"PCA_196"**$\}$ the output of **'min_U'** are 5 matrices of size 1X1. In other cases the output are 5 matrices of size 5X6. Each entry of the matrix corresponds to the minimum value of W_u at the best epoch, for a specific hyperparameter ε and N_{pt}.

The keys **'mean_diameter_U'**, **'mean_diameter_P'** are used to plot Figures 10 and 11 in the paper.

- **All_confusion_matrices.p** This pickled dictionary contains confusion matrices, accuracies, and (in the case of binary classifiers) F1 scores.

  * It's first key is so that
  $$\text{key}_1 \in \{ \text{"Classifier\_196"}, \text{"PCA\_196"} \}. \tag{18}$$
  * Its 2nd and 3rd keys are as in (5) and (6), respectively.
  * A 4th key is
  $$\text{key}_4 \in \{ \text{'raw'}, \text{'accuracy'}, \text{'relative'}, \text{'f1\_score'} \} \tag{19}$$
  The outputs for **'accuracy'** and **'f1_score'** are somewhat clear. The key **'raw'** corresponds to a confusion matrix with absolute values, while **'relative'** gives values as a proportion over the number of true labels (the sum over rows of each confusion matrix).

For more examples, see (Monteiro, 2021b, PSBC_ensemble_learning_notebook.ipynb).

- **All_confusion_matrices_multiclass.p** Contains the confusion matrices for the multiclassifier described in Section 3.2 of the paper.

  * It's first key takes a single value,
  $$\text{key}_1 \in \{ \text{"Classifier\_196"} \}. \tag{20}$$
  * Its second keys is as in (5).
  * Its 3rd key corresponds to the voting system used during the one-versus-one method. It is
  $$\text{key}_3 \in \{ \text{'hard\_vote'}, \text{'tournament'} \} \tag{21}$$
  Hard vote is standard, while 'tournament' corresponds to several rounds of hard voting, where the predicted label with the lowest score is eliminated, followed by a new round that only considers the remaining individuals. The vote is then assigned as explained in the footnote 6 in the paper.
  * Its 4th key is
  $$\text{key}_4 \in \{ \text{'relative'}, \text{'raw'} \}. \tag{22}$$
  As explained in the case of **All_confusion_matrices.p**

For more examples, see (Monteiro, 2021b, PSBC_ensemble_learning_notebook.ipynb).

where **folder_name in {"Neumann", "Periodic", "Neumann_non_subordinate", "Classifier_196"}** and **(variable_0,variable_1)** indicates the pair of distinct digits associated to that weight.

2.6 Content in Pickled_datasets

In the folder MNIST you will find six other pickled files. The first two that we discuss are

- **X_A_B.p**, where $A \in \{\textbf{train}, \textbf{test}\}$ and $B \in \{0, \ldots, 9\}$.
- **generate_k_fold_a_b.p**, where a, b are distinct digits in $\{0, \ldots, 9\}$, always following the convention that $a < b$. This pickled dictionaries are used in cross validation. They are obtained as the output of files **mnist_pickled.py** (see the beginning of part II).

  Since in Section 3.4 we are comparing PSBC models with different boundary conditions, we split the folds used during cross validation in the same way, reducing the difference is results to models' differences rather than occasional statistical differences that different folds could offer.

  Due to the use of PCA to construct basis matrices in Section 3.3 in the paper, these dictionaries also contain principal components associated with each fold. These matrices are obtained by standard Singular Value Decomposition.

Part III
Examples, jupyter-notebooks, and reproducibility

3. Data & reproducibility

All the following files are available for download at Monteiro (2021a), as tarballs. The only folders necessary for replicating the data are represented in a different color in the tree 1. They are compressed as three tarballs:

- The folder **Pickled_datasets** is by far the biggest of them (about 500 Mb), containing the MNIST digits split as required in the paper. It is available as **PSBC_dataset.tar.gz**.
- The main modules, scripts, and supporting files — all in **MOTHER_PSBC**, **Grids**, and **Statistics** — are available in **PSBC_libs_grids_statistics.tar.gz**. They can also be downloaded at Monteiro (2021b).
- All jupyter notebooks are available as **PSBC_notebooks.tar.gz**. They can also be downloaded at Monteiro (2021b).

The remaining tarballs contains the data generated in our tests, featured in the Section 3 of the paper. Some of the jupyter-notebooks can be used to reproduce the figures that summarize such data (see Section 4 below).

- The data generated in the study of different boundary conditions —contained in the folders **Neumann**, **Periodic**, and **Neumann_non_subordinate** — are available at **PSBC_BCs.tar.gz**. They can also be downloaded at Monteiro (2021b).
- The data generated for the study of the multiclass classifier and classifier with PCA — folders **Classifier_196** and **PCA_196** — are available as **PSBC_classifier_PCA.tar.gz**. They can also be downloaded at Monteiro (2021b).

4. Jupyter notebooks

All the jupyer-notebooks used in this project have to be placed in the folder **PSBC**. Initially, some of these notebooks were split because they had to either run in different machines or be submitted to super computers using PBS scripts, which required python script versions of them. Nevertheless, all these programs had a similar structure, the main reason why they were redesigned into single jupyter-notebooks - one for grid searching, another for training. With them, all the studies shown in Section 3 of the paper can be reproduced.

We purposefully exploit the design of the folder tree 1 and its symmetries. As such, using a single notebook speeds up iterations, tests, besides allowing model selection and model assessment

to take place almost seamlessly. By and large, output files have the same name, although we break this symmetry sometimes in order to short circuit the program in case any loading of weights to the incorrect models is attempted.

It is not hard to convert these jupyter-notebooks into python scripts that can be run in a supercomputer, as we did in order to train the PSBC with different boundary conditions.

A brief description of each jupyter-notebook is given below.

- **PSBC_grid_search_notebook.ipynb** is responsible for grid search. The accessed folder needs to be defined by the user.
- **PSBC_training_notebook.ipynb** is used for training (model assessment). The accessed folder needs to be defined by the user.
- **PSBC_ensemble_learning.ipynb** retrieves pickled dictionaries in the folder **Statistics** and plot associated confusion matrices.
- **PSBC_using_statistical_files.ipynb** explains how statistics can be retrieved. It also retrieves pickled dictionaries in the folder **Statistics** and plot surfaces of accuracies (in terms of parameters N_t, N_{pt}, ε, and layers-k-shared). It also generates some plots seem in the Supplementary Material.
- **PBC_examples_notebook.ipynb** contains several examples of how the model can be used, how its layers can be saved (or retrieved from h5 files), and be optimized.

References

Rafael Monteiro. Data repository for the paper "Binary classification as a phase separation process". https://dx.doi.org/10.5281/zenodo.4005131, September 2021a.

Rafael Monteiro. Source code for the paper "Binary classification as a phase separation process". https://github.com/rafael-a-monteiro-math/Binary_classification_phase_separation, September 2021b.