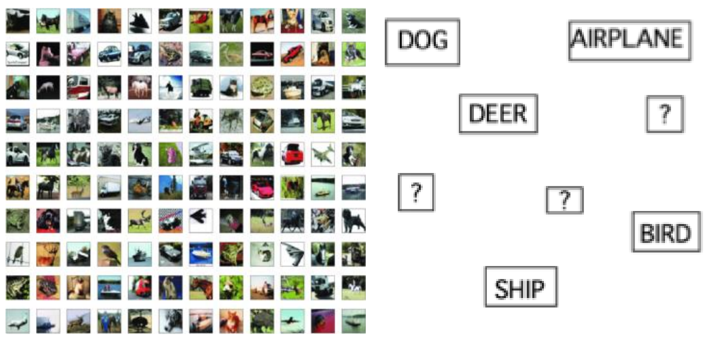
SemiPy: a simple Semi-Supervised Learning Python Library

Main developer: Lucas Boiteau
Joint work with:
Alexandre Gensse, Quentin Oliveau, Hugo Schmutz, Aude Sportisse, Pierre-Alexandre Mattei

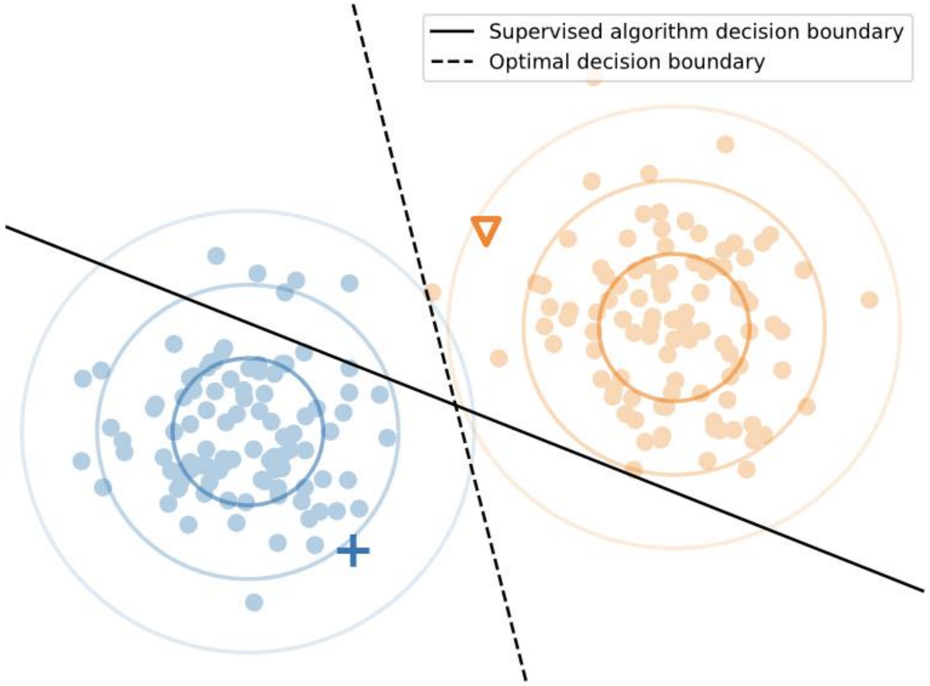
Sophia Summit, November 24 2023, Biot

Semi-supervised learning (SSL)



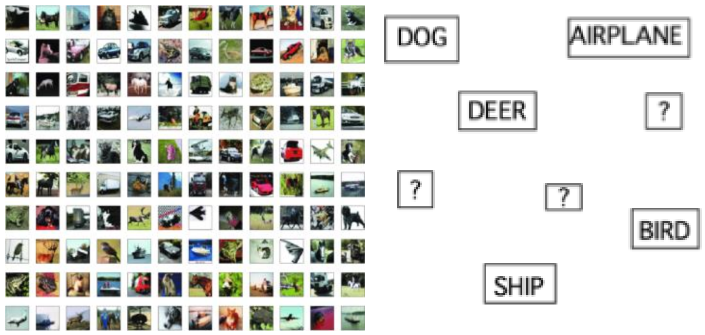
- Context: huge amount of data available, but labelling the data is costly and time-consuming.

- Goal: using both **labelled and unlabelled data** to build predictive models.



V.Engelen et al., 2020

Mathematical setting



- n iid samples:

$$D = \{(x_i, y_i)\}_{i=1}^{n}$$

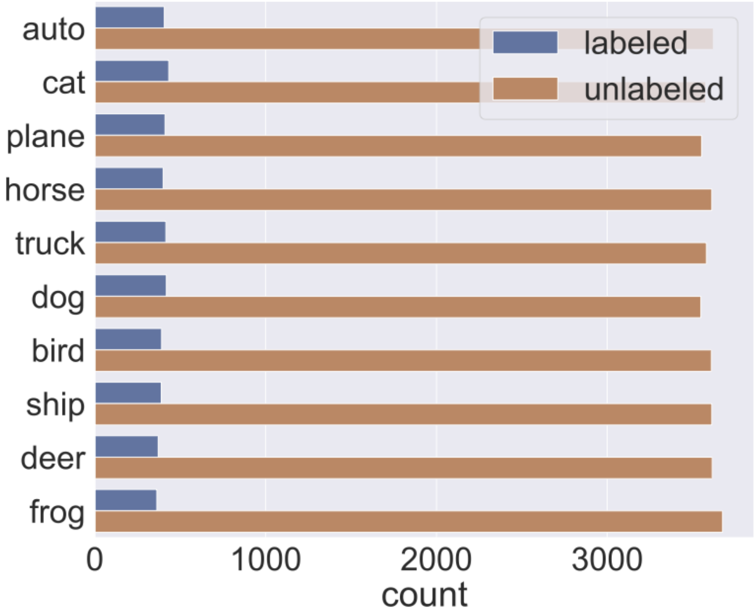
Features   Labels

- We observe n_l labelled data and n_u unlabelled data:

$$D_l = \{(x_i, y_i)\}_{i=1}^{n_l} \qquad D_u = \{(x_i)\}_{i=n_l+1}^{n}$$

- We want to learn $p(y|x;\theta)$

What we consider:



- **Deep** SSL: $p(y|x;\theta)$ is a neural network.
- **Classification task**: $y \in \{0, \dots, K\}$
- **Missing Completely At Random (MCAR) assumption:** the label distributions are identical in the labeled and unlabeled dataset

It implies that the **ratio unlabeled-labeled data is the same for each class**.

Classical SSL approach

- Goal: minimize the risk to learn $p(y|x;\theta)$
- *Supervised* empirical risk: $\hat{R}\,(\theta) := \frac{1}{n}\sum_{i=1}^{n} L(\theta; x_i, y_i)$ **NOT TRACTABLE**
- Complete-case empirical risk:

$$\hat{R}^{cc}(\theta) := \frac{1}{n_l}\sum_{i=1}^{n_l} L(\theta; x_i, y_i)$$

ONLY ON THE LABELED DATA

Classical SSL approach

- SSL empirical risk:

$$\hat{R}^{SSL}(\theta) := \frac{1}{n_l}\sum_{i=1}^{n_l} L(\theta; x_i, y_i) + \lambda \frac{1}{n_u}\sum_{i=n_l+1}^{n} H(\theta; x_i)$$

TERM ON THE LABELED DATA **+λ** **TERM ON THE UNLABELED DATA**

- $\lambda > 0$ is the regularization parameter
- Choice of $H(\theta; x_i)$: in many cases it is a surrogate of $L(\theta; x_i, y_i)$
  - Entropy minimization:

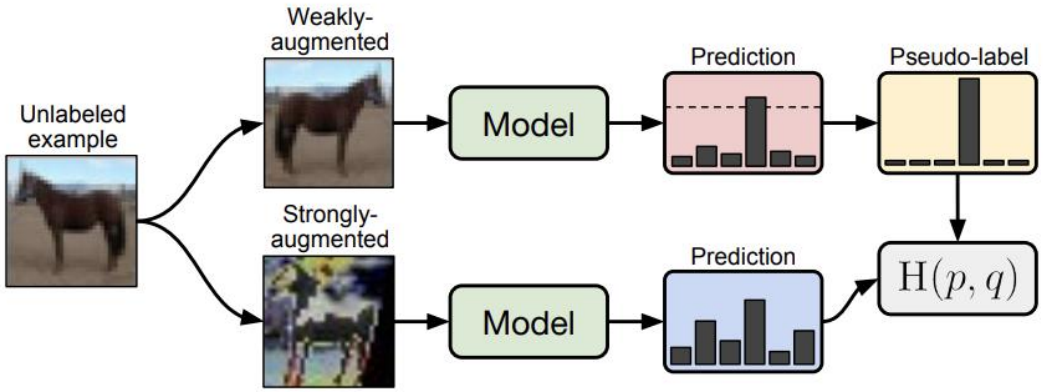
$$H(\theta; x_i) = E[L(\theta; x_i, y_i) | x_i]$$

Two classical SSL methods

- **Pseudo-labels:**
  - choose the class c with the maximum predicted probability
  - only the pseudo-labels which have a maximum predicted probability larger than a predefined threshold τ are used as target:

$$H(\theta; x) = -\log p(c|x;\theta)1_{\max p(y|x;\theta)>\tau}$$

- **Fixmatch:** Robustness of the model to data augmentation of the features



SemiPy

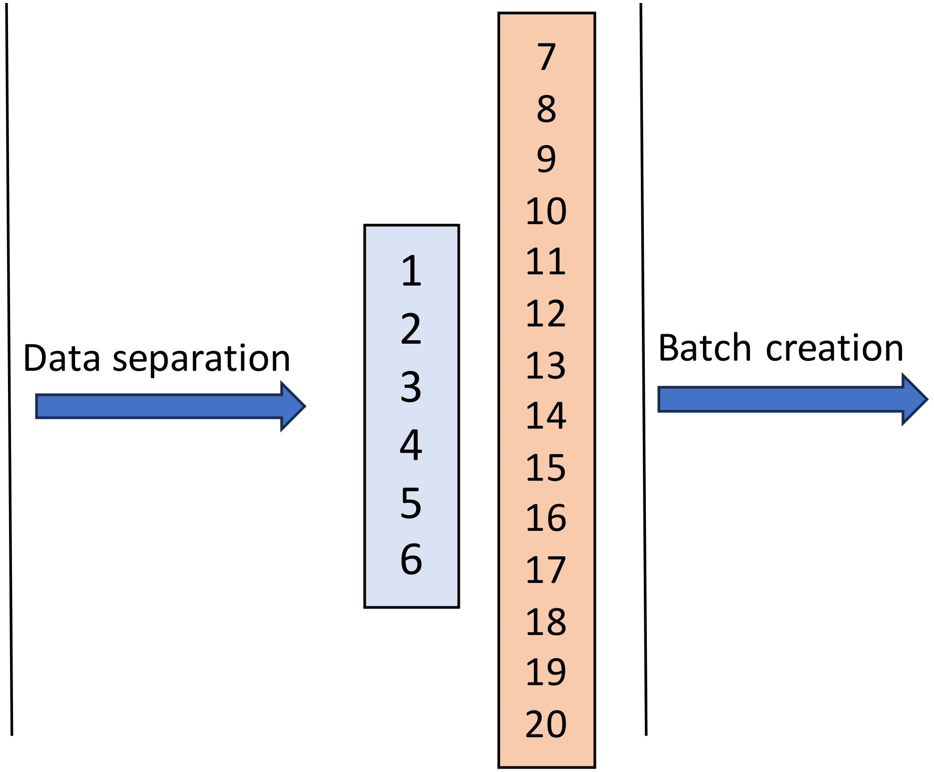
- Open-source Python library
- Toolbox for Semi-Supervised Learning
- Including most famous SSL algorithms and datasets
- Modular library that can be adapted and extended

Key features

- Sampler : a useful batch sampler that allows to use only one dataloader for both labelled and unlabelled data

```
array([[ 1,  1],
       [ 2,  1],
       [ 3,  1],
       [ 4,  2],
       [ 5,  3],
       [ 6,  3],
       [ 7, -1],
       [ 8, -1],
       [ 9, -1],
       [10, -1],
       [11, -1],
       [12, -1],
       [13, -1],
       [14, -1],
       [15, -1],
       [16, -1],
       [17, -1],
       [18, -1],
       [19, -1],
       [20, -1]])
```

Fake data




```
----------Epoch 0----------
tensor([1, 2, 7, 8], dtype=torch.int32)
tensor([ 3,  4,  9, 10], dtype=torch.int32)
tensor([ 5,  6, 11, 12], dtype=torch.int32)
----------Epoch 1----------
tensor([ 6,  3, 13, 14], dtype=torch.int32)
tensor([ 5,  1, 15, 16], dtype=torch.int32)
tensor([ 2,  4, 17, 18], dtype=torch.int32)
----------Epoch 2----------
tensor([ 4,  5, 19, 20], dtype=torch.int32)
tensor([ 6,  1, 13, 10], dtype=torch.int32)
tensor([ 3,  2, 18, 19], dtype=torch.int32)
----------Epoch 3----------
tensor([ 2,  6, 17,  7], dtype=torch.int32)
tensor([1, 3, 9, 8], dtype=torch.int32)
tensor([ 5,  4, 11, 12], dtype=torch.int32)
----------Epoch 4----------
tensor([ 3,  4, 20, 14], dtype=torch.int32)
tensor([ 1,  2, 16, 15], dtype=torch.int32)
tensor([ 5,  6, 13,  9], dtype=torch.int32)
```

Labelled samples have been shuffled

Unlabelled samples have been shuffled

Training simulation

L. Boiteau, A. Gensse, Q. Oliveau, H. Schmutz, A. Sportisse, P.-A. Mattei

Key features

- Versatility : either using a configuration file or more in-depth functions to create your own workflow in a script or a Notebook

python™

PyTorch

- Configuration file usage

```yaml
USE_LIGHTNING: True
EPOCHS: 1
BALANCING_WEIGHT: 0.5
DEBIASED: False
SELECTION_THRESHOLD: 0.95
BATCH_SIZE: 64
LABELLED_PROPORTION: 0.5
SAVE_PATH: './saves'
OPTIMIZER:
  NAME: 'SGD'
  PARAMS:
    lr: 1.0e-3
    momentum: 0.9
SCHEDULER: null
NET: 'resnet18'
METHOD: 'pseudolabel'
NUM_WARMUP_EPOCHS: null
DATA:
  NAME: null
  VALIDATION_PROPORTION: null
  TEST_PROPORTION: null
  LABELLED_SAMPLES: null
  UNLABELLED_SAMPLES: null
  INCLUDE_LABELLED: True
  USE_EXTRA: False # Used by SVHN dataset
  SPLITS:
    TRAIN:
      PATH: 'data'
      NAME_UNLABELLED: 'nodata'
      TRANSFORMS: []
```

```yaml
USE_MULTIGPU: False
NUM_GPU: null
MULTIGPU_STRATEGY: null
EMA: null
METRICS:
  VALIDATION:
    - NAME: Accuracy
      PARAMS:
        task: multiclass
  TEST:
    - NAME: Accuracy
      PARAMS:
        task: multiclass
EARLYSTOPPING:
  NAME: VALIDATION/Loss
  PARAMS:
    mode: min
    patience: 10
```

- Notebook/script usage with built-in functions

```python
import semipy as smp
trainer = smp.tools.SSLTrainer(config='config.yaml')
trainer.fit()
```

```
Files already downloaded and verified
Files already downloaded and verified
EPOCH 6: 6% 6/100 [02:38<38:41, 24.70s/it, Epoch_Loss=1.6, Last_Validation_Loss=2.38]
Iterations: 25% 16/63 [00:05<00:16, 2.83it/s, Loss=1.62]
```

- Access to all loss functions and sampler

```python
loss_fn = smp.methods.FixMatchLoss(model=model, lbda=0.5, threshold=0.95, debiased=False)
```

```python
sets = smp.datasets.get_cifar(name='cifar10', num_labelled=4000)
sampler = smp.sampler.JointSampler(sets['Train'], batch_size=64, proportion=0.5)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

- Script usage

```
$ python main.py --config config.yaml
```

Key features

- Customization ability:
  - Use your own dataset
  - Use your own model
  - Easily add a new SSL method
  - In configuration file: easily add metrics and data augmentation

```yaml
METRICS:
  VALIDATION:
    - NAME: Accuracy
      PARAMS:
        task: multiclass
  TEST:
    - NAME: Accuracy
      PARAMS:
        task: multiclass
```

Adding metrics

```yaml
CUSTOM:
  WEAK_TRANSFORM:
    - NAME: RandomHorizontalFlip
      PARAMS:
        p: 0.5
    - NAME: ToTensor
      PARAMS: {}

  STRONG_TRANSFORM:
    - NAME: RandAugment
      PARAMS:
        num_ops: 3
    - NAME: ToTensor
      PARAMS: {}
```

Adding data augmentation

Key features

- PyTorch Lightning support
- GPU and multi-GPU support
- Debiased Semi-Supervised Learning

Schmutz, H., Humbert, O., & Mattei, P. A. (ICLR 2023). Don't fear the unlabelled: safe deep semi-supervised learning via simple debiasing.

```python
from pytorch_lightning import Trainer
import semipy as smp
trainer = Trainer(max_epochs=100, accelerator='gpu', check_val_every_n_epoch=10)
lightning_module = smp.pl.LitFixMatch(config='config.yaml')
trainer.fit(lightning_module)
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
Files already downloaded and verified
Files already downloaded and verified
You are using a CUDA device ('NVIDIA RTX A2000 8GB Laptop GPU') that has Tensor Cores. To properly utilize them, you should set `torch.set_float32_matmul_precision('medium' | 'high')` which will trade-off precision for performance. For more details, read https://pytorch.org/docs/stable/generated/torch.set_float32_matmul_precision.html#torch.set_float32_matmul_precision
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type       | Params
---------------------------------------
0 | model   | ResNet     | 11.2 M
1 | metrics | ModuleDict | 0
---------------------------------------
11.2 M    Trainable params
0         Non-trainable params
11.2 M    Total params
44.727    Total estimated model params size (MB)
Epoch 3: 21%  13/63 [00:40<02:37, 3.15s/it, v_num=3]
```


L. Boiteau, A. Gensse, Q. Oliveau, H. Schmutz, A. Sportisse, P.-A. Mattei

Perspectives

- More datasets, more algorithms, more functionalities (metrics per class)
- Beyond image datasets: text and audio data
- Beyond the MCAR assumption, when there are some **popular classes**

Sportisse, A., Schmutz, H., Humbert, O., Bouveyron, C., & Mattei, P. A. (ICML 2023). Are labels informative in semi-supervised learning? Estimating and leveraging the missing-data mechanism

- Involving the SSL community more

https://semipy.github.io