Main concepts

Connect is an application to train and evaluate machine learning models on distributed datasets without centralising the data and more generally without compromising the privacy of the data. Connect keeps a traceability of operations done to train and evaluate models.

It supports simple training and evaluation scheme, such as training a model on data in a center A and evaluating the model on data in a center B, or more complex federated learning (FL) schemes, such as Federated Averaging. It also enables to do both horizontal FL (same features across participants) and vertical FL (different features across participants), as well as multi-partners multi-task learning (multi tasks that can be different from one participant to another).

To make this possible, 4 main assets are defined:

- Metrics
- Datasets and Data Samples
- Algorithms
- Models and Compute Plans

Metrics
A metric is used to evaluate a model performance. Concretely, a metric corresponds to an - archive (tar or zip file) containing
- Python scripts which code for the metric computation
- a Dockerfile which contains the required dependencies of the Python scripts

Dataset:
A dataset represents the data in Connect. It is made up of:
- an opener, which is a script used to load the data from files into memory.
- Data samples: a folder containing the data files

Algorithms
An algorithm specifies the method to train a model on a dataset. It specifies the model type and architecture, the loss function, the optimizer, hyperparameters and, also identifies the parameters that are tuned during training. Concretely, an algorithm corresponds to an archive (tar or zip file) containing:
- Python scripts which code for the algorithm. Importantly, a train and a predict functions have to be defined
- a Dockerfile which contains the required dependencies of the Python scripts

There are three types of algorithms:

- Simple algorithm:

This algorithm can be used with tasks of kind train tuple and produces a single model.

- Composite algorithm:

This kind of algorithm makes it possible to train a trunk and head model:

- the trunk model can be shared among all nodes
- the head model remains private to the node where it was trained

This enables vertical FL, as well as multi-partners multi-task FL.

- Aggregate algorithm

This algorithm type is used to aggregate models or model updates. An aggregate algorithm does not need data to be used.

Models/Model updates

A Model or a Model Update is a potentially large file containing the parameters or update of parameters of a trained model. In the case of a neural network, a model would contain the weights of the connections. It is either the result of training an Algorithm with a given Dataset, corresponding to a training task (Train tuple or Composite train tuple); or the result of an Aggregate Algorithm aggregating models or model updates; corresponding to an aggregation task (Aggregate tuple).

Compute plans and tasks

Compute plan

A set of training (Train tuple or Composite train tuple), aggregation (Aggregate tuple) and testing tasks (Test tuple) gathered together towards building a final model.
The user should register a compute plan rather than separate tasks.

Train tuple

The specification of a training task of a simple algorithm on a dataset potentially using input models or model updates. It leads to the creation of a model or model update.
Schéma inputs > outputs

Composite train tuple

The specification of a training task of a composite algorithm on a dataset potentially using input trunk and head models or model updates. It leads to the creation of a trunk and head model or model update. Depending on associated permissions, a trunk model or model update can be shared with other nodes, whereas a head model remains in the node where it was created.
Schéma inputs > outputs

Aggregate tuple

The specification of an aggregation task of several models or model updates using an aggregate algo. It leads to the creation of one model or model update.
Schéma inputs > outputs

Test tuple

The specification of a testing task of a model. It evaluates the performance of the model using one or several metrics with a dataset.
Schéma inputs > outputs

Permissions

Permissions are set on the following assets:

- dataset
- algorithm
- metric
- trunk model created by a composite train tuple

Dataset, algorithm, metric

In the following table, the asset is registered by nodeA with the permissions {public: False, authorized_ids: [nodeA, nodeB]}

Asset	Node	What can this node and its users do?
Dataset	nodeA	● nodeA can do any task on this dataset, on nodeA ● nodeA users can export the dataset's opener
	nodeB	● nodeB can do any task on this dataset, on nodeA ● nodeB users can export the dataset's opener
	nodeC	● nodeC cannot do anything on this dataset ● nodeC users cannot export the dataset's opener
Algo	nodeA	● nodeA can use the algo in a task on any node (if it also has the permissions on the other assets used in the task) ● nodeA users can export the algo's archive
	nodeB	● nodeB can use the algo in a task on any node (if it also has the permissions on the other assets used in the task) ● nodeB users can export the algo's archive
	nodeC	● nodeC cannot use the algo in a task whatever the node ● nodeC users cannot export the algo's archive
Metric	nodeA	● nodeA can use the metric in a test task on any node (if it also has the permissions on the other assets used in the task) ● nodeA users can export the metric's archive
	nodeB	● nodeB can use the metric in a test task on any node (if it also has the permissions on the other assets used in the task) ● nodeB users can export the metric's archive
	nodeC	● nodeC cannot use the metric in a test task whatever the node (to check with the new metric object as this was not true with the objective object) ● nodeC users cannot export the metric's archive

Model (out-model)

There are no permissions on train / aggregate / composite train / test tasks.

There are permissions on the out-models of the train / aggregate / composite train tasks.

NodeA has the permissions on the out-model of a task if:

- for a train / composite train task, nodeA has permissions on the algo and on the dataset used in the task
- for an aggregate task, nodeA has permissions on the aggregate algo and on all the models used in the task

NodeA has permissions on the out-model of a task means nodeA can use this model in any type of task on any node (if it also has the permissions on the other assets used in the task).

The user of nodeA can export an out-model with the CLI/SDK if:

- nodeA has the permissions on the out-model
- nodeA has the variable model_export_enabled set to True (at the level of the node within the channel)

Specificity of trunk and head out-models in a composite train task

- The head out-model permissions are set to be non-public, and processable/downloadable only by the associated dataset's owner
- The trunk out-model permissions are specified explicitly by the user when registering the composite train task