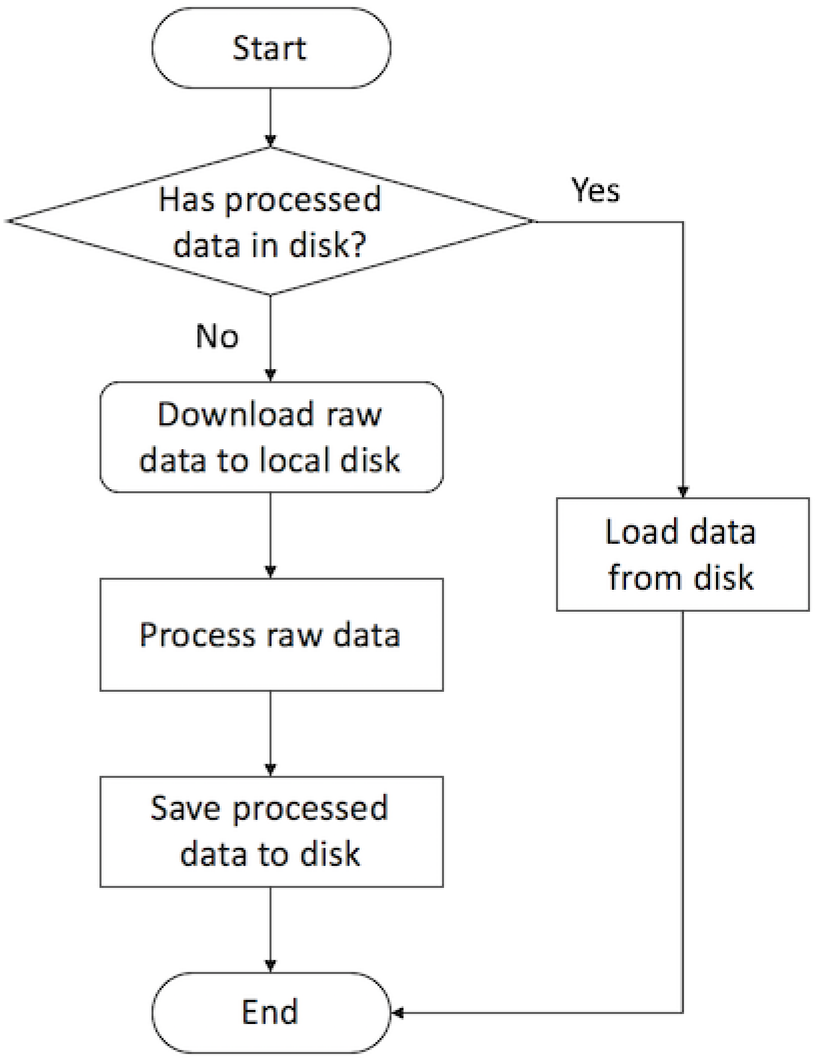
4.1 DGLDataset class

(中文版)

`DGLDataset` is the base class for processing, loading and saving graph datasets defined in dgl.data. It implements the basic pipeline for processing graph data. The following flow chart shows how the pipeline works.

To process a graph dataset located in a remote server or local disk, one can define a class, say `MyDataset`, inheriting from `dgl.data.DGLDataset`. The template of `MyDataset` is as follows.


Start
Has processed data in disk?
Yes
No
Download raw data to local disk
Process raw data
Save processed data to disk
Load data from disk
End

Flow chart for graph data input pipeline defined in class DGLDataset.

```python
from dgl.data import DGLDataset

class MyDataset(DGLDataset):
    """ Template for customizing graph datasets in DGL.

    Parameters
    ----------
    url : str
        URL to download the raw dataset
    raw_dir : str
        Specifying the directory that will store the
        downloaded data or the directory that
        already stores the input data.
        Default: ~/.dgl/
    save_dir : str
        Directory to save the processed dataset.
        Default: the value of `raw_dir`
    force_reload : bool
        Whether to reload the dataset. Default: False
    verbose : bool
        Whether to print out progress information
    """
    def __init__(self,
                 url=None,
                 raw_dir=None,
                 save_dir=None,
                 force_reload=False,
                 verbose=False):
        super(MyDataset, self).__init__(name='dataset_name',
                                        url=url,
                                        raw_dir=raw_dir,
                                        save_dir=save_dir,
                                        force_reload=force_reload,
                                        verbose=verbose)

    def download(self):
        # download raw data to local disk
        pass

    def process(self):
        # process raw data to graphs, labels, splitting masks
        pass

    def __getitem__(self, idx):
        # get one example by index
        pass

    def __len__(self):
        # number of data examples
        pass

    def save(self):
        # save processed data to directory `self.save_path`
        pass

    def load(self):
        # load processed data from directory `self.save_path`
        pass

    def has_cache(self):
        # check whether there are processed data in `self.save_path`
        pass
```

`DGLDataset` class has abstract functions `process()`, `__getitem__(idx)` and `__len__()` that must be implemented in the subclass. DGL also recommends implementing saving and loading as well, since they can save significant time for processing large datasets, and there are several APIs making it easy (see 4.4 Save and load data).

Note that the purpose of `DGLDataset` is to provide a standard and convenient way to load graph data. One can store graphs, features, labels, masks and basic information about the dataset, such as number of classes, number of labels, etc. Operations such as sampling, partition or feature normalization are done outside of the `DGLDataset` subclass.

The rest of this chapter shows the best practices to implement the functions in the pipeline.