OnClass

Release OnClass0.0.1

Aug 13, 2021

Contents

OnClass is a python package for single-cell cell type annotation. It uses the Cell Ontology to capture the cell type similarity. These similarities enable OnClass to annotate cell types that are never seen in the training data.

CHAPTER 1

Introduction

OnClass is a python package for single-cell cell type annotation. It uses the Cell Ontology to capture the cell type similarity. These similarities enable OnClass to annotate cell types that are never seen in the training data.

A preprint of OnClass paper is on bioRxiv All datasets used in OnClass can be found in figshare. Currently, OnClass supports

1) annotate cell type
2) integrating different single-cell datasets based on the Cell Ontology
3) marker genes identification

OnClass is a joint work by Altman lab at stanford and czbiohub.

For questions about the software, please contact Sheng Wang at swang91@uw.edu.

Our web server can be found at: https://onclass.readthedocs.io/.

CHAPTER 2

Install OnClass

OnClass can be substantially accelerated by using GPU (tensorflow 2.0). However, this is only required when you want to train your own model. OnClass can also be used with only CPU.

OnClass package only has three scripts: the OnClass class file, the deep learning model file, and the utility functions file.

You can simply get these three files and put it in the right directory (see how to use run_OnClass_example.py in Tutorial). You can also install OnClass using pip as following:

1) Only use CPU

```
pip install OnClass==1.2
pip install tensorflow==2.0
```

2) Use GPU

```
pip install OnClass==1.2
pip install tensorflow-gpu==2.0
```

2.1 Development Version

The lastest verion of OnClass is on GitHub

```
git clone https://github.com/wangshenguiuc/OnClass.git
```

CHAPTER 3

Dataset and pretrained models

If you want to use your own training data, please use OnClass_data_public_minimal.tar.gz. OnClass_data_public_minimal.tar.gz includes the minimal files (ontology files) needed to run OnClass on your own dataset. You need to provide the annotated cells (training data) and then OnClass can classify unannotated cells.

If you don't have your own training data, you can use the annotated gene expression data (e.g., Tabula Muris Senis, Lemur, HLCA, Allen brian) used in OnClass paper or pretrained models. See scRNA_data below for how to download the annotated gene expression data. See Pretrained_model below for how to download the pretrained model.

3.1 1) scRNA_data

Download three parts from link 1, link 2, link 3. Jointly extract the files using

```
cat OnClass_data_public_scRNA_data.tar.gz.* | tar -xz
```

This will give you all the single cell gene expression data used in our paper (see Fig. 2, Extended Data Figs. 1-3, Supplementary Figs. 4-7).

3.2 2) Ontology_data

These files are in OnClass_data_public_minimal.tar.gz. They include Cell Ontology and Allen brain Ontology. Cell Ontology has cell type text definition. cl.ontology.nlp.emb is the text embedding of the definition of each cell type.

3.3 3) Pretrained_model

Download 8 tensorflow pretrained models here. They are trained from 8 dataset in Fig. 2, Extended Data Figs. 1-3, Supplementary Figs. 4-7.

3.4 4) Intermediate_files

This folder contains the intermediate files. Data generated by example scripts will be stored here.

We suggest you organize all the downloaded files as the following:

```
.
├── Intermediate_files
│   ├── Cross_dataset [31 entries exceeds filelimit, not opening dir]
│   └── Marker_genes [22 entries exceeds filelimit, not opening dir]
├── Ontology_data
│   ├── allen.ontology
│   ├── cl.obo
│   ├── cl.ontology
│   └── cl.ontology.nlp.emb
├── Pretrained_model [37 entries exceeds filelimit, not opening dir]
└── scRNA_data
    ├── 26-datasets
    │   ├── 293t_jurkat
    │   ├── brain
    │   ├── hsc
    │   ├── macrophage
    │   ├── pancreas
    │   └── pbmc
    ├── Allen_Brain
    │   ├── features.pkl
    │   ├── genes.pkl
    │   └── labels.pkl
    ├── gene_marker_expert_curated.txt
    ├── HLCA
    │   ├── 10x_features.pkl
    │   ├── 10x_genes.pkl
    │   └── 10x_labels.pkl
    ├── Lemur
    │   ├── microcebusAntoine.h5ad
    │   ├── microcebusBernard.h5ad
    │   ├── microcebusMartine.h5ad
    │   └── microcebusStumpy.h5ad
    └── Tabula_Muris_Senis
        ├── tabula-muris-senis-droplet-official-raw-obj.h5ad
        └── tabula-muris-senis-facs-official-raw-obj.h5ad
```

For questions about the datasets, please contact Sheng Wang at swang91@uw.edu.

CHAPTER 4

How to run OnClass

To run OnClass, please first install OnClass, download datasets and then change file paths in config.py

We provide a run_OnClass_example.py and Jupyter notebook as an example to run OnClass. This script trains an OnClass model on all cells from one Lemur dataset, saves that model to a model file, then use this model to classify cells from another Lemur dataset.

4.1 Run your own dataset for cell type annotation

You only need to modify line 9-13 in run_OnClass_example.py by replacing train_file, test_file with your training and test file, and train_label and test_label with the cell ontology label key in your dataset.

Import OnClass and other libs as:

```
from anndata import read_h5ad
from scipy import stats, sparse
import numpy as np
import sys
from collections import Counter
from OnClass.OnClassModel import OnClassModel
from utils import read_ontology_file, read_data, run_scanorama_multiply_datasets
from config import ontology_data_dir, scrna_data_dir, model_dir, Run_scanorama_batch_
↪correction, NHIDDEN, MAX_ITER
```

Read training and test data. Set nlp_mapping = True to use the Char-level LSTM that maps uncontrolled vocabulary to controlled vocabulary. If you don't want to use h5ad file, you can provide training and test data in the format of numpy array to OnClass. Training and test features (gene expression) should be cell by gene 2D array. Training label should be a vector of cell labels.

```
train_file = scrna_data_dir + '/Lemur/microcebusBernard.h5ad'
test_file = scrna_data_dir + '/Lemur/microcebusAntoine.h5ad'

train_label = 'cell_ontology_id'
```

(continues on next page)

(continued from previous page)

```
test_label = 'cell_ontology_id'
model_path = model_dir + 'example_file_model'

cell_type_nlp_emb_file, cell_type_network_file, cl_obo_file = read_ontology_file(
↪'cell ontology', ontology_data_dir)
OnClass_train_obj = OnClassModel(cell_type_nlp_emb_file = cell_type_nlp_emb_file,␣
↪cell_type_network_file = cell_type_network_file)

train_feature, train_genes, train_label, _, _ = read_data(train_file, cell_ontology_
↪ids = OnClass_train_obj.cell_ontology_ids,
        exclude_non_leaf_ontology = False, tissue_key = 'tissue', AnnData_label_key =␣
↪train_label, filter_key = {},
        nlp_mapping = False, cl_obo_file = cl_obo_file, cell_ontology_file = cell_
↪type_network_file, co2emb = OnClass_train_obj.co2vec_nlp)
```

Embed the cell ontology:

```
OnClass_train_obj.EmbedCellTypes(train_label)
```

Batch correction using Scanorama:

```
if Run_scanorama_batch_correction:
        train_feature, test_feature = run_scanorama_multiply_datasets([train_feature,␣
↪test_feature], [train_genes, test_genes], scan_dim = 10)[1]
```

Training:

```
cor_train_feature, cor_test_feature, cor_train_genes, cor_test_genes = OnClass_train_
↪obj.ProcessTrainFeature(train_feature, train_label, train_genes, test_feature =␣
↪test_feature, test_genes = test_genes)
OnClass_train_obj.BuildModel(ngene = len(cor_train_genes), nhidden = NHIDDEN)
OnClass_train_obj.Train(cor_train_feature, train_label, save_model = model_path, max_
↪iter = MAX_ITER)
```

Test:

```
OnClass_test_obj = OnClassModel(cell_type_nlp_emb_file = cell_type_nlp_emb_file, cell_
↪type_network_file = cell_type_network_file)
cor_test_feature = OnClass_train_obj.ProcessTestFeature(cor_test_feature, cor_test_
↪genes, use_pretrain = model_path, log_transform = False)
OnClass_test_obj.BuildModel(ngene = None, use_pretrain = model_path)

pred_Y_seen, pred_Y_all, pred_label = OnClass_test_obj.Predict(cor_test_feature, test_
↪genes = cor_test_genes, use_normalize=True)
pred_label_str = [OnClass_test_obj.i2co[l] for l in pred_label]
```

4.2 One dataset cross-validation

run_one_dataset_cross_validation.py can be used to reproduce Figure 2 in our paper. All data are provided in figshare (please see Dataset and pretrained model)

4.3 Cross dataset prediction

run_cross_dataset_prediction.py can be used to reproduce Figure 4 in our paper. All data are provided in figshare (please see Dataset and pretrained model)

4.4 Marker genes identification

Please first run run_generate_pretrained_model.py to generate the intermediate files (line 53-54) for marker gene prediction.

Train a model using the seen cell types:

```
OnClass_train_obj.EmbedCellTypes(train_label)
print ('generate pretrain model. Save the model to $model_path...')
model_path = model_dir + 'OnClass_full_'+dname
train_feature, train_genes = OnClass_train_obj.ProcessTrainFeature(train_feature,␣
↪train_label, train_genes)
OnClass_train_obj.BuildModel(ngene = len(train_genes))
OnClass_train_obj.Train(train_feature, train_label, save_model = model_path)
```

Use this model to classify cells into all cell types in the Cell Ontology. Here pred_Y_seen is a cell by seen cell type matrix, pred_Y_all is a cell by all cell type type matrix.

```
OnClass_test_obj = OnClassModel(cell_type_nlp_emb_file = cell_type_nlp_emb_file, cell_
↪type_network_file = cell_type_network_file)
OnClass_test_obj.BuildModel(ngene = None, use_pretrain = model_path)
pred_Y_seen, pred_Y_all, pred_label = OnClass_test_obj.Predict(train_feature, test_
↪genes = train_genes, use_normalize=False, use_unseen_distance = -1)
np.save(output_dir+dname + 'pred_Y_seen.released.npy',pred_Y_seen)
np.save(output_dir+dname + 'pred_Y_all.released.npy',pred_Y_all)
```

Then run run_marker_genes_identification.py for marker gene identification (Figure 5c).

Run run_marker_gene_based_prediction.py for marker gene based prediction (Figure 5d,e,f, Extended Data Figure 7).

CHAPTER 5

References

Bibliography

[Hie19] Brian Hie, Bryan Bryson, and Bonnie Berger. *Efficient integration of heterogeneous single-cell transcriptomes using Scanorama.*, Nature biotechnology 37.6 (2019): 685.

[Wang19] Sheng Wang, Angela Oliveira Pisco, Jim Karkanias, and Russ B. Altman. *Unifying single-cell annotations based on the Cell Ontology.*, bioRxiv