scDeepSort: A Pre-trained Cell-type Annotation Method for Single-cell Transcriptomics using Deep Learning with a Weighted Graph Neural Network

Recent advance in single-cell RNA sequencing (scRNA-seq) has enabled large-scale transcriptional characterization of thousands of cells in multiple complex tissues, in which accurate cell type identification becomes the prerequisite and vital step for scRNA-seq studies.

To addresses this challenge, we developed a pre-trained cell-type annotation method, namely scDeepSort, using a state-of-the-art deep learning algorithm, i.e. a modified graph neural network (GNN) model. In brief, scDeepSort was constructed based on our weighted GNN framework and was then learned in two embedded high-quality scRNA-seq atlases containing 764,741 cells across 88 tissues of human and mouse, which are the most comprehensive multiple-organs scRNA-seq data resources to date.

For more information, please refer to a preprint in bioRxiv.
1.1 Getting Started

This is a quick start guide for you to try out scDeepSort. The full script is available at script.

1.1.1 Evaluate with Pre-trained Models

Define the Model

scDeepSort provides unified APIs on evaluating different datasets with pre-trained models.
For the demo on cell type annotating, the corresponding model is DeepSortPredictor:

```python
from deepsort import DeepSortPredictor
model = DeepSortPredictor(species='human', tissue='Spleen')
```

Currently we support cell type annotation on human and mouse datasets, with available tissues listed in GitHub Wiki Page (human and mouse). Note that the first letter of tissues should be in upper case.

Prepare Data

Please refer to Input Requirement
Evaluate

Once the datasets prepared, users can predict the corresponding cell type (and subtype if exists) for cells. Our predict function supports processing single file in one pass as following:

```python
test_file = 'test/human/human_Spleen11081_data.csv'
predictor.predict(test_file, save_path='results')
```

This method saves results to specific path if provided as keyword argument.

The default setting on hyper-parameters enables scDeepSort to perform reasonably well across all datasets. Please refer to API Reference for the meaning of different input arguments.

1.1.2 2. Train Your Own Model

Below is the full script on using scDeepSort for classification on a demo dataset.

```python
from deepsort import DeepSortClassifier

# define the model
model = DeepSortClassifier(species='human',
                            tissue='Brain',
                            dense_dim=50,
                            hidden_dim=20,
                            gpu_id=0,
                            n_layers=2,
                            random_seed=1,
                            n_epochs=20)

train_files = [('/path/to/human_brain_data_1.csv', '/path/to/human_brain_celltype_1.csv'),
               ('/path/to/human_brain_data_2.csv', '/path/to/human_brain_celltype_2.csv')]

test_files = ['/path/to/human_brain_test_data_1.csv', '/path/to/human_brain_test_data_2.csv']

# fit the model
model.fit(train_files, save_path='model_save_path')

# use the saved model to predict
for test_file in test_files:
    model.predict(test_file, save_path='results', model_path='model_save_path')
```

Our DeepSortClassifier model takes a list of tuples of file paths as inputs to fit on multiple datasets.

Users are required to prepare the data file and the corresponding cell type file for training and testing as expected in Input Requirement.

1.2 Installation Guide

1.2.1 Stable Version

scDeepSort requires Python version 3.7, 3.8 or 3.9.

For the convenience of users, we provide CPU and CUDA builds in different compressed files. These builds share the same package name.

For users with GPU support, please first check out your CUDA version by running the following command:
For users with a CUDA 10.2 build, please download scDeepSort-v1.0-cu102.tar.gz from our GitHub release page. The package is then available using:

```
$ pip install scDeepSort-v1.0-cu102.tar.gz
```

The package is used a few package dependencies which will be handled automatically. It is recommended to use the package environment from Anaconda since it already installs all required packages. Notice that only the 64-bit Linux are officially supported.

### 1.2.2 Quick Start

To verify your installation, you can run the following test code.

```python
import deepsort
deepest.data.demo()
```

The program will end with “Test successfully!” if everything is good.

### 1.3 Input Requirement

#### 1.3.1 Data File

scDeepSort assumes data to be in the form of 2D table of shape \((\text{num\_of\_genes}, \text{num\_of\_cells})\) with index and header included. For example:

<table>
<thead>
<tr>
<th></th>
<th>Cell 1</th>
<th>Cell 2</th>
<th>...</th>
<th>Cell N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene 1</td>
<td>0</td>
<td>2.4</td>
<td>...</td>
<td>5.0</td>
</tr>
<tr>
<td>Gene 2</td>
<td>0.8</td>
<td>1.1</td>
<td>...</td>
<td>4.3</td>
</tr>
<tr>
<td>Gene 3</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Gene M</td>
<td>1.8</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

We recommend csv format for input data matrix. The input data matrix should be pre-processed by first revising gene symbols according to NCBI Gene Database updated on Jan. 10, 2020, wherein unmatched genes and duplicated genes will be removed. Then it should be normalized with the default \(\text{LogNormalize}\) method in Seurat (R package), detailed in R Script, wherein the column represents each cell and the row represent each gene for training and testing data, as shown above.

scDeepSort will conduct internal check and transformation on the input data according to given data paths.
1.3.2 Cell Type File

We assume csv format for all cell type files. The csv file should have three columns including index columns, cell name and its corresponding cell type. It is only used in DeepSortClassifier.fit.

```
"","Cell","Cell_type"
"1","Cell 1","Conventional CD4+ T cell"
"2","Cell 2","Conventional CD4+ T cell"
...
"N","Cell N","Conventional CD4+ T cell"
```

1.4 API Reference

Below is the class and function reference for scDeepSort. Notice that the package is under active development, and some features may not be stable yet.

1.4.1 DeepSortPredictor

```
# Class Definition
DeepSortPredictor(species,
    tissue,
    file_type='csv',
    unsure_rate=2.)
```

- **species**: The species of cells, human or mouse.
- **tissue**: The tissue of cells. For the detailed list of supported tissue of our model, please refer to GitHub Wiki page.
- **file_type**: The format of data file, csv or gz. csv for .csv files and gz for .gz files. For details, please refer to R script.
- **unsure_rate**: The multiplier for unsure threshold (computed as unsure_rate / num_classes) to cast the type of cell to the unsure type, default to 2.0. Set it as 0 to exclude the unsure type.

```
# Class Method
DeepSortPredictor.predict(input_file,
    save_path=None) -> pandas.DataFrame
```

- **input_file**: The file path for test dataset.
- **save_path**: The destination for saving predictions. Save results to disk if path provided.

Example

```python
from deepsort import DeepSortPredictor
# define the model
model = DeepSortPredictor(species='human',
    tissue='Brain')
# use the trained model to predict
test_files = ['/path/to/human_brain_test_data_1.csv', '/path/to/human_brain_test_data_2.csv']
for test_file in test_files:
    model.predict(test_file, save_path='results', model_path='model_save_path')
```
### Class Definition

```python
DeepSortClassifier(species,  
    tissue,  
    dense_dim=400,  
    hidden_dim=200,  
    batch_size=256,  
    dropout=0.1,  
    gpu_id=-1,  
    file_type='csv',  
    learning_rate=0.001,  
    weight_decay=5e-4,  
    n_epochs=300,  
    n_layers=1,  
    threshold=0,  
    num_neighbors=None,  
    exclude_rate=0.005,  
    random_seed=None,  
    validation_fraction=0.1)
```

- **species**: The species of cells, human or mouse.
- **tissue**: The tissue of cells. For the detailed list of supported tissue of our model, please refer to GitHub Wiki page.
- **dense_dim**: The initial dimension of node embedding for cells and genes. Default to 400.
- **hidden_dim**: The hidden dimension of Weighted Graph Aggregator Layer. Default to 200.
- **batch_size**: The number of samples per batch. Default to 256.
- **dropout**: The dropout rate for the output representation of Weighted Graph Aggregator Layer, default to 0.1.
- **gpu_id**: The GPU id for training and testing. -1 for CPU. Default to -1.
- **file_type**: The format of data file, csv or gz. csv for .csv files and gz for .gz files.
- **learning_rate**: The learning rate of optimizer. Default to 0.001.
- **weight_decay**: The weight decay of optimizer. Default to 0.00004.
- **n_epochs**: Maximum number of epochs. Default to 300.
- **n_layers**: The number of layers. Default to 1.
- **threshold**: The weight threshold for edges between cell nodes and gene nodes. Default to 0.
- **num_neighbors**: The number of sampled neighbors per nodes in training time. If None, all the neighbors will be sampled.
- **exclude_rate**: Exclude a class if the portion of this class is less than exclude_rate. Default to 0.005.
- **random_seed**: For reproducibility. Fixed if given. Default to None.
- **validation_fraction**: The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Default to 0.1.

### Class Method

```python
DeepSortClassifier.fit(files,  
    save_path=None)
```

- **files**: The file path for training datasets. We assume files in the form of (data_file, celltype_file).
• **save_path**: The destination for saving models.

```python
# Class Method
DeepSortClassifier.predict(input_file,
    model_path,
    save_path=None,
    unsure_rate=2.,
    file_type='csv') -> pandas.DataFrame
```

• **input_file**: The file path for test dataset.

• **model_path**: The path for loading saved models.

• **save_path**: The destination for saving predictions. Save results to disk if path provided.

**Example**

```python
from deepsort import DeepSortClassifier
# define the model
model = DeepSortClassifier(species='human',
    tissue='Brain',
    dense_dim=50,
    hidden_dim=20,
    gpu_id=0,
    n_layers=2,
    random_seed=1,
    n_epochs=20)

train_files = [('/path/to/human_brain_data_1.csv', '/path/to/human_brain_celltype_1.csv'),
               ('/path/to/human_brain_data_2.csv', '/path/to/human_brain_celltype_2.csv')]

test_files = ['/path/to/human_brain_test_data_1.csv', '/path/to/human_brain_test_data_2.csv']

# fit the model
model.fit(train_files, save_path='model_save_path')
# use the saved model to predict
for test_file in test_files:
    model.predict(test_file, save_path='results', model_path='model_save_path')
```

### 1.5 About Us

scDeepSort manuscript is under major revision. Should you have any questions, please contact

- Xin Shao at xin_shao@zju.edu.cn
- Haihong Yang at capriceyhh@zju.edu.cn
- Xiang Zhuang at zhuangxiang@zju.edu.cn