vipcca

Release 0.2.7

Jialu Hu

Aug 25, 2023

CONTENTS

Data alignment is one of the first key steps in single cell analysis for integrating multiple datasets and performing joint analysis across studies. Data alignment is challenging in extremely large datasets, however, as the major of the current single cell data alignment methods are not computationally efficient. Here, we present VIPCCA, a computational framework based on non-linear canonical correlation analysis for effective and scalable single cell data alignment. VIPCCA leverages both deep learning for effective single cell data modeling and variational inference for scalable computation, thus enabling powerful data alignment across multiple samples, multiple data platforms, and multiple data types. VIPCCA is accurate for a range of alignment tasks including alignment between single cell RNAseq and ATACseq datasets and can easily accommodate millions of cells, thereby providing researchers unique opportunities to tackle challenges emerging from large-scale single-cell atlas.

CHAPTER

ONE

VIPCCA INSTALLATION

1.1 Installation

- Create conda environment

```
$ conda create -n vipcca python=3.8
$ conda activate vipcca
```

- Install vipcca from pypi

```
$ pip install vipcca
```

- Install vipcca from GitHub source code

```
$ git clone https://github.com/jhu99/vipcca.git
$ cd ./vipcca/
$ pip install .
```

Note: Please make sure your python version >= 3.7. The current release depends on tensorflow with version 2.4.0. Install tenserfolow-gpu if gpu is avialable on the machine.

CHAPTER

TWO

VIPCCA TUTORIAL

2.1 Tutorials for VIPCCA

2.1.1 Integrating multiple scRNA-seq data

This tutorial shows loading, preprocessing, VIPCCA integration and visualization of 293T and Jurkat cells in three different batches (Mixed Cell Lines).

Importing vipcca package

Here, we'll import vipcca along with other popular packages.

[1]:
```
import vipcca.model.vipcca as vip
import vipcca.tools.utils as tl
import vipcca.tools.plotting as pl

import matplotlib
import scanpy as sc
matplotlib.use('TkAgg')

# Command for Jupyter notebooks only
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

```
Using TensorFlow backend.
```

Loading data

This tutorial uses Mixed Cell Line datasets from 10xgenomics with non-overlapping populations from three batches, two of which contain 293t (2885 cells) and jurkat (3258 cells) cells respectively, and the third batch contains a 1:1 mixture of 293t and jurkat cells (3388 cells).

- Read from 10x mtx file The file in 10x mtx format can be downloaded here. Set the fmt parameter of pp.read_sc_data() function to '10x_mtx' to read the data downloaded from 10XGenomics. If the file downloaded from 10XGenomics is in h5 format, the dataset can be loaded by setting the fmt parameter to '10x_h5'.

[2]:
```
base_path = "./data/vipcca/mixed_cell_lines/"
file1 = base_path+"293t/hg19/"
file2 = base_path+"jurkat/hg19/"
file3 = base_path+"mixed/hg19/"

adata_b1 = tl.read_sc_data(file1, fmt='10x_mtx', batch_name="293t")
adata_b2 = tl.read_sc_data(file2, fmt='10x_mtx', batch_name="jurkat")
adata_b3 = tl.read_sc_data(file3, fmt='10x_mtx', batch_name="mixed")
```

- Read from h5ad file The h5ad file we generated that containing the cell type can be downloaded here. Here, we load the three datasets separately.

[3]:
```
base_path = "./data/vipcca/mixed_cell_lines/"

adata_b1 = tl.read_sc_data(base_path+"293t.h5ad", batch_name="293t")
adata_b2 = tl.read_sc_data(base_path+"jurkat.h5ad", batch_name="jurkat")
adata_b3 = tl.read_sc_data(base_path+"mixed.h5ad", batch_name="mixed")
```

Each dataset is loading into an AnnData object.

[4]:
```
adata_b3
```

[4]:
```
AnnData object with n_obs × n_vars = 3388 × 32738
    obs: '_batch', 'celltype'
    var: 'gene_ids'
```

Data preprocessing

Here, we filter and normalize each data separately and concatenate them into one AnnData object. For more details, please check the preprocessing API.

[5]:
```
adata_all = tl.preprocessing([adata_b1, adata_b2, adata_b3], index_unique="-")
```

```
Trying to set attribute `.obs` of view, copying.
Trying to set attribute `.obs` of view, copying.
Trying to set attribute `.obs` of view, copying.
```

VIPCCA Integration

[6]:
```
# Command for Jupyter notebooks only
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

# The four hyperparameters epochs, lambda_regulizer, batch_input_size, batch_input_
↪size2 are user-defined parameters.
# Given a dataset with ~10K cells, we suggest to pick up
# a value larger than 600 for epochs,
# a value in the range of [2,10] for lambda_regulizer,
# a value at least less than the number of input features for batch_input_size,
# a value in the range of [8,16] for batch_input_size2.
handle = vip.VIPCCA(
    adata_all=adata_all,
    res_path='./data/vipcca/mixed_cell_lines/',
    mode='CVAE',
    split_by="_batch",
```

(continues on next page)

(continued from previous page)

```
    epochs=20,
    lambda_regulizer=5,
    batch_input_size=128,
    batch_input_size2=16
)
# do integration and return an AnnData object
adata_integrate = handle.fit_integrate()
```

```
WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling␣
↪BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)␣
↪with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Model: "vae_mlp"
_________________________________________________________________________________
↪_______________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================
↪===============
encoder_input (InputLayer)   (None, 2000)              0
_________________________________________________________________________________
↪_______________
batch_input1 (InputLayer)    (None, 128)               0
_________________________________________________________________________________
↪_______________
encoder_mlp (Model)          [(None, 16), (None,  318368      encoder_input[0][0]
                                                              batch_input1[0][0]
_________________________________________________________________________________
↪_______________
batch_input2 (InputLayer)    (None, 16)                0
_________________________________________________________________________________
↪_______________
decoder_mlp (Model)          (None, 2000)              546272      encoder_mlp[1][2]
                                                              batch_input2[0][0]
=================================================================================
↪===============
Total params: 864,640
Trainable params: 862,496
Non-trainable params: 2,144
_________________________________________________________________________________
↪_______________
WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.
↪python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables␣
↪is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/keras/callbacks/tensorboard_v1.py:200: The name tf.summary.merge_all␣
↪is deprecated. Please use tf.compat.v1.summary.merge_all instead.

WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/keras/callbacks/tensorboard_v1.py:203: The name tf.summary.FileWriter␣
↪is deprecated. Please use tf.compat.v1.summary.FileWriter instead.
```

(continues on next page)

(continued from previous page)

```
Epoch 1/20
9530/9530 [==============================] - 4s 382us/step - loss: 2928.3410
WARNING:tensorflow:From /Users/zhongyuanke/anaconda3/envs/test-vipcca/lib/python3.7/
↪site-packages/keras/callbacks/tensorboard_v1.py:343: The name tf.Summary is␣
↪deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/20
9530/9530 [==============================] - 1s 112us/step - loss: 2569.9079
Epoch 3/20
9530/9530 [==============================] - 1s 109us/step - loss: 2555.9494
Epoch 4/20
9530/9530 [==============================] - 1s 108us/step - loss: 2535.0038
Epoch 5/20
9530/9530 [==============================] - 1s 108us/step - loss: 2528.3331
Epoch 6/20
9530/9530 [==============================] - 1s 105us/step - loss: 2520.0686
Epoch 7/20
9530/9530 [==============================] - 1s 106us/step - loss: 2516.1301
Epoch 8/20
9530/9530 [==============================] - 1s 106us/step - loss: 2508.4249
Epoch 9/20
9530/9530 [==============================] - 1s 108us/step - loss: 2500.8367
Epoch 10/20
9530/9530 [==============================] - 1s 107us/step - loss: 2494.5631
Epoch 11/20
9530/9530 [==============================] - 1s 108us/step - loss: 2491.3073
Epoch 12/20
9530/9530 [==============================] - 1s 108us/step - loss: 2486.6252
Epoch 13/20
9530/9530 [==============================] - 1s 110us/step - loss: 2480.4250
Epoch 14/20
9530/9530 [==============================] - 1s 109us/step - loss: 2480.4351
Epoch 15/20
9530/9530 [==============================] - 1s 113us/step - loss: 2480.8911
Epoch 16/20
9530/9530 [==============================] - 1s 115us/step - loss: 2476.3571
Epoch 17/20
9530/9530 [==============================] - 1s 114us/step - loss: 2475.4372
Epoch 18/20
9530/9530 [==============================] - 1s 114us/step - loss: 2472.8237
Epoch 19/20
9530/9530 [==============================] - 1s 115us/step - loss: 2470.7086
Epoch 20/20
9530/9530 [==============================] - 1s 115us/step - loss: 2468.2385
... storing '_batch' as categorical
... storing 'celltype' as categorical
```

[7]:
```
adata_integrate
```

[7]:
```
AnnData object with n_obs × n_vars = 9530 × 2000
    obs: '_batch', 'celltype', 'n_genes', 'percent_mito', 'n_counts', 'size_factor',
↪'batch'
    var: 'gene_ids', 'n_cells-0-0', 'highly_variable-0-0', 'means-0-0', 'dispersions-
↪0-0', 'dispersions_norm-0-0', 'n_cells-1-0', 'highly_variable-1-0', 'means-1-0',
↪'dispersions-1-0', 'dispersions_norm-1-0', 'n_cells-1', 'highly_variable-1', 'means-
↪1', 'dispersions-1', 'dispersions_norm-1'
    obsm: 'X_vipcca'
```

1.The meta.data of each cell has been saved in adata.obs

2.The embedding representation from vipcca of each cell have been saved in adata.obsm('X_vipcca')

Loading result

- **Loading result from saved model.h5 file through vipcca**: The model.h5 file of the trained result can be downloaded here

[8]:
```
model_path = 'model.h5'
handle = vip.VIPCCA(
    adata_all=adata_all,
    res_path='./data/vipcca/mixed_cell_lines/',
    mode='CVAE',
    split_by="_batch",
    epochs=20,
    lambda_regulizer=5,
    batch_input_size=128,
    batch_input_size2=16,
    model_file=model_path
)
adata_integrate = handle.fit_integrate()
```

```
Model: "vae_mlp"
_________________________________________________________________________________________
↪_____________
Layer (type)                 Output Shape              Param #     Connected to
=========================================================================================
encoder_input (InputLayer)   (None, 2000)              0
_________________________________________________________________________________________
↪_____________
batch_input1 (InputLayer)    (None, 128)               0
_________________________________________________________________________________________
↪_____________
encoder_mlp (Model)          [(None, 16), (None,  318368      encoder_input[0][0]
                                                              batch_input1[0][0]
_________________________________________________________________________________________
↪_____________
batch_input2 (InputLayer)    (None, 16)                0
_________________________________________________________________________________________
↪_____________
decoder_mlp (Model)          (None, 2000)              546272      encoder_mlp[1][2]
                                                              batch_input2[0][0]
=========================================================================================
Total params: 864,640
Trainable params: 862,496
Non-trainable params: 2,144
_________________________________________________________________________________________
↪_____________
```

```
... storing '_batch' as categorical
... storing 'celltype' as categorical
```

- **Loading result from h5ad file**: The output.h5ad file of the trained result can be downloaded here

[9]:
```
integrate_path = './data/vipcca/mixed_cell_line/output.h5ad'
adata_integrate = sc.read_h5ad(integrate_path)
```

UMAP Visualization

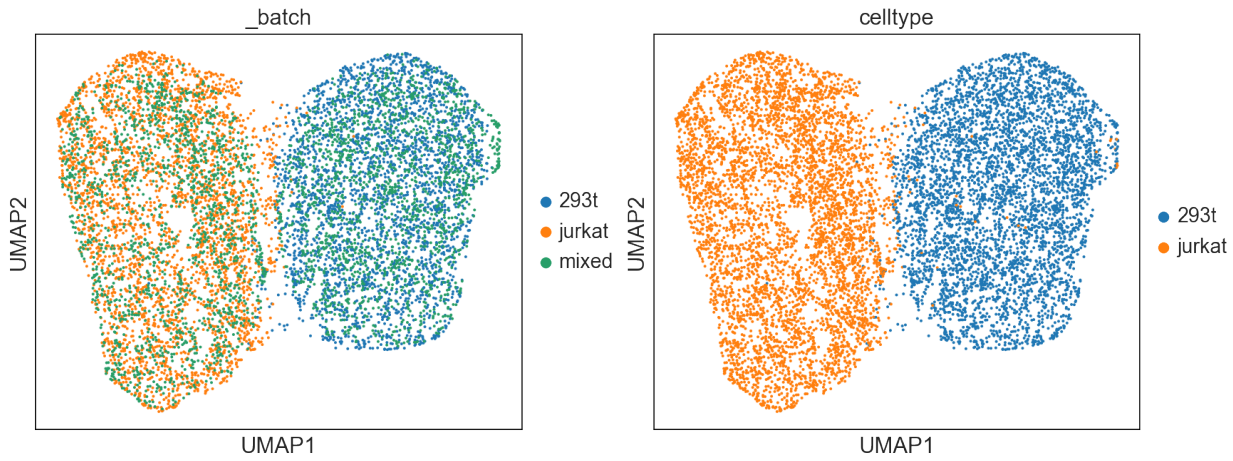
We use UMAP to reduce the embedding feature output by vipcca in 2 dimensions.

[8]:
```
sc.pp.neighbors(adata_integrate, use_rep='X_vipcca')
sc.tl.umap(adata_integrate)
```

Visualization of UMAP result.

[9]:
```
sc.set_figure_params(figsize=[5.5,4.5])
sc.pl.umap(adata_integrate, color=['_batch','celltype'], use_raw=False, show=True,)
```

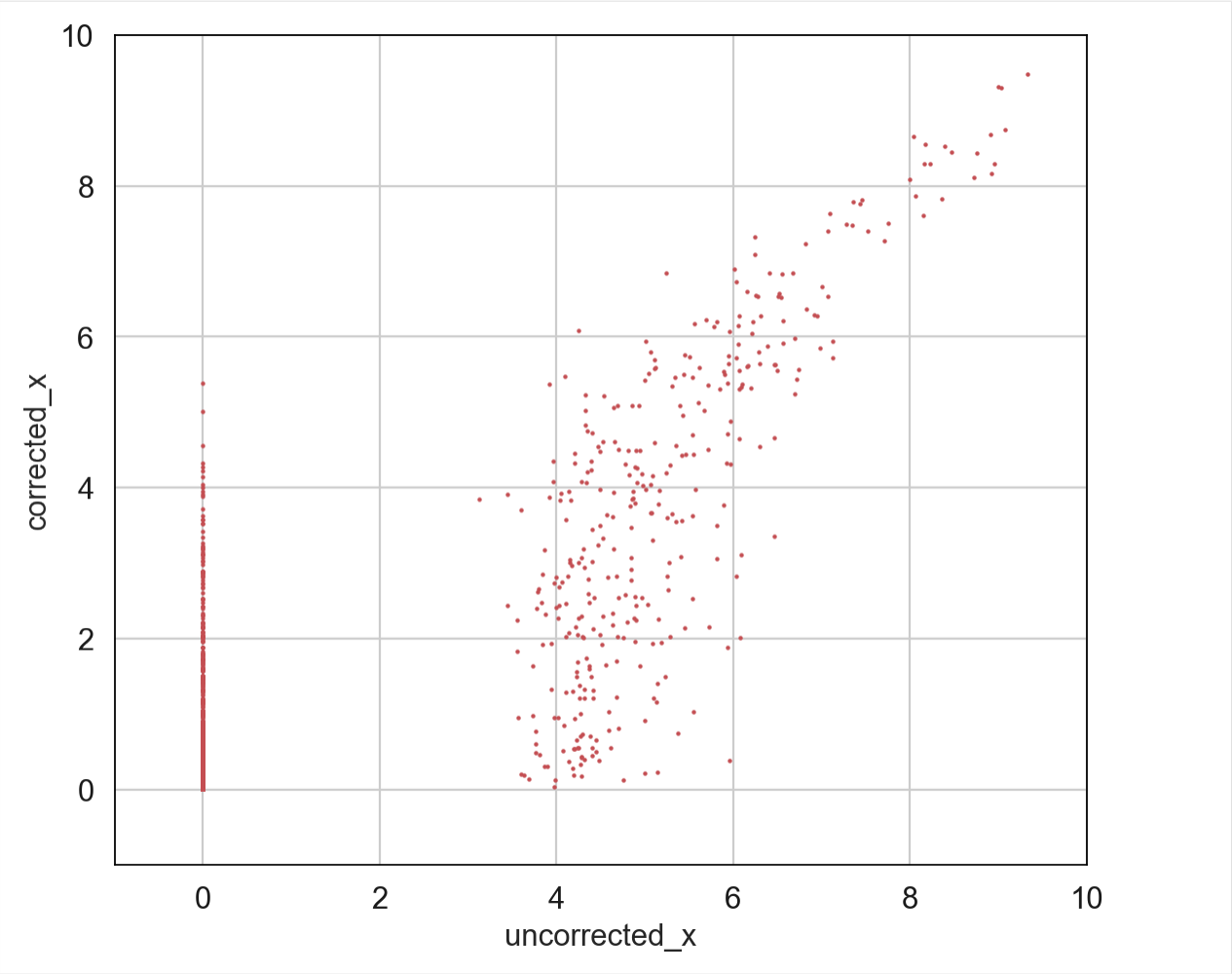
```
... storing '_batch' as categorical
... storing 'celltype' as categorical
```



Plot correlation

This function is only applicable to AnnData generated by fit_integrate() function training. Randomly select 2000 locations in the gene expression matrix. The x-axis represents the expression value of the original data at these locations, and the y-axis represents the expression value of the data after vipcca integration at the same location.

[10]:
```
pl.plotCorrelation(adata_integrate.raw.X, adata_integrate.X, save=False, rnum=2000,␣
↪lim=10)
```



2.1.2 Analysis of VIPCCA integrated result in R

We implement downstream analysis based on the R language and some R packages (Seurat, KBET), including evaluating the degree of mixing and gene differential expression analysis.

Required packages

Several R packages need to be installed in the following analysis, including Seurat, SeuratDisk, kBET.

Read h5ad file with R

Import packages

```
library(Seurat)
library(SeuratDisk)
```

Convert h5ad files to h5seurat files After executing this code, a file in h5seurat format will be generated in the path where the h5ad file is located

```
Convert("/Users/zhongyuanke/data/vipcca/mixed_cell_lines_result/output_save.h5ad",␣
↪dest = "h5seurat", overwrite = TRUE)
```

Read h5seurat file into a Seurat Object

```
mcl <- LoadH5Seurat("/Users/zhongyuanke/data/vipcca/mixed_cell_lines_result/output_
↪save.h5seurat")
```

Calculating kBET

Preparing kBET data

```
celltype <- t(data.frame(mcl@meta.data[["celltype"]]))
vipcca_emb <- data.frame(mcl@reductions[["vipcca"]]@cell.embeddings)
batch <- mcl@meta.data[["X_batch"]]

# Split the batch ID and VIPCCA embedding result by cell type.
vipcca_emb <- split(vipcca_emb,celltype)
batch <- split(batch,celltype)
```

Calculating kBET for 293T celltype.

For detailed usage of kbet, please check the kBET tutorial .

```
library(kBET)

subset_size <- 0.25 #subsample to 25% of the data.
subset_id <- sample.int(n = length(t(batch[["293t"]])), size = floor(subset_size *␣
↪length(t(batch[["293t"]]))), replace=FALSE)
batch.estimate_1 <- kBET(data_mix[["293t"]][subset_id,], batch[["293t"]][subset_id])
```

Differential gene analyses

We evaluate the results of integration by analyzing the differential expression genes between different batches. For more detail, see the documentation of FindMarkers() function.

First, we read the h5seurat file into a Seurat object.

```
mcl <- LoadH5Seurat("/Users/zhongyuanke/data/vipcca/mixed_cell_lines_result/output_
↪save.h5seurat")
```

We use 293T cells from batches of '293t' and 'mixed as an example'.

```
library(Seurat)

Idents(mcl) <- 'celltype'
mcl$celltype.cond <- paste(Idents(mcl), mcl@meta.data[['X_batch']], sep = "_")
Idents(mcl) <- "celltype.cond"

br <- FindMarkers(mcl, ident.1 = '293t', ident.2 = 'mixed',
                  slot = "data",
                  logfc.threshold = 0.,
                  test.use = "wilcox",
                  verbose = FALSE,
                  min.cells.feature = 1,
                  min.cells.group = 1,
                  min.pct = 0.1)

boxplot(br$p_val_adj,
        main = "Adjusted P-value for each gene",
        xlab = "293T",
        ylab = "Adjusted P-value")
```

Plotting Enhanced Volcano

Volcano plots represent a useful way to visualise the results of differential expression analyses. The smaller the number of differentially expressed genes between two batches, the better the effect of batch effect removal. For more detail, see the documentation of EnhancedVolcano.

```
library(EnhancedVolcano)

EnhancedVolcano(br,
                lab = rownames(br),
                x = 'avg_log2FC',
                y = 'p_val_adj',
                title = 'Volcano plot for 293T',
                )
```

2.1.3 scRNA-seq+scATAC-seq integration

This tutorial shows loading, preprocessing, VIPCCA integration and visualization of scRNA-seq and scATAC-seq dataset.

we obtained a scRNA-seq data consisting of gene expression measurements on 33,538 genes in 11,769 cells and a scATAC-seq data consisting of 89,796 open chromatin peaks on 8,728 nuclei, both were produced by 10X Genomics Chromium system and were on PBMCs.

Preprocessing with R

For the scRNA-seq data: Seurat have previously pre-processed and clustered a scRNA-seq dataset and annotate 13 celltype, and provide the object here. The 13 cell types include 460 B cell progenitor, 2,992 CD14+ Monocytes, 328 CD16+ Monocytes, 1,596 CD4 Memory, 1,047 CD4 Naïve, 383 CD8 effector, 337 CD8 Naïve, 74 Dendritic cell, 592 Double negative T cell, 544 NK cell, 68 pDC, 52 Plateletes, and 599 pre-B cell.

For the scATAC-seq data: First, we load in the provided peak matrix and collapse the peak matrix to a "gene activity matrix" using Signac. Then we filtered out cells that have with fewer than 5,000 total peak counts to focus on a final set of 7,866 cells for analysis. See the Signac website for tutorial and documentation for analysing scATAC-seq data.

After preprocessing, you can converting Seurat Object to AnnData via h5Seurat using R packages (SeuratDisk). In this case, the atac.h5ad file will be generated in the corresponding path.

```
library(SeuratDisk)
SaveH5Seurat(atac, filename = "atac.h5Seurat")
Convert("atac.h5Seurat", dest = "h5ad")
```

Besides, you can also directly download the processed scRNA-seq dataset and scATAC-seq dataset files in H5ad format.

Importing vipcca package

[1]:
```
import vipcca.model.vipcca as vip
import vipcca.tools.utils as tl
import vipcca.tools.plotting as pl
import vipcca.tools.transferLabel as tfl
import scanpy as sc
from sklearn.preprocessing import LabelEncoder

import matplotlib
matplotlib.use('TkAgg')

# Command for Jupyter notebooks only
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

Loading data in python

[2]:
```
adata_rna = tl.read_sc_data("./rna.h5ad")
adata_atac = tl.read_sc_data("./geneact.h5ad")
```

Data preprocessing

Here, we filter and normalize each data separately and concatenate them into one AnnData object. For more details, please check the preprocessing API.

[3]:
```
adata_all= tl.preprocessing([adata_rna, adata_atac])
```

```
Trying to set attribute `.obs` of view, copying.
Trying to set attribute `.obs` of view, copying.
```

VIPCCA Integration

[4]:
```
# Command for Jupyter notebooks only
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

# The four hyperparameters epochs, lambda_regulizer, batch_input_size, batch_input_
↪size2 are user-defined parameters.
# Given a dataset with ~10K cells, we suggest to pick up
# a value larger than 600 for epochs,
# a value in the range of [2,10] for lambda_regulizer,
# a value at least less than the number of input features for batch_input_size,
# a value in the range of [8,16] for batch_input_size2.
handle = vip.VIPCCA(adata_all=adata_all,
                            res_path='./atac_result/',
                            mode='CVAE',
                            split_by="_batch",
                            epochs=20,
                            lambda_regulizer=2,
                            batch_input_size=64,
                            batch_input_size2=14,
                            )
adata_integrate=handle.fit_integrate()
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify␣
↪the frequency in number of batches seen.
Model: "vae_mlp"
_____
↪_____
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
encoder_input (InputLayer)   [(None, 2000)]            0
_____
↪_____
batch_input1 (InputLayer)    [(None, 64)]              0
_____
↪_____
encoder_mlp (Functional)     [(None, 16), (None,  285088      encoder_input[0][0]
                                                              batch_input1[0][0]
_____
↪_____
```

(continues on next page)

(continued from previous page)

```
batch_input2 (InputLayer)       [(None, 14)]         0
_________________________________________________________________
↪_______________
decoder_mlp (Functional)        (None, 2000)         542748      encoder_mlp[0][2]
                                                                 batch_input2[0][0]
_________________________________________________________________
↪_______________
dense (Dense)                   (None, 64)           4160        batch_input1[0][0]
_________________________________________________________________
↪_______________
batch_normalization (BatchNorma (None, 64)           192         dense[0][0]
_________________________________________________________________
↪_______________
activation (Activation)         (None, 64)           0           batch_
↪normalization[0][0]
_________________________________________________________________
↪_______________
dense_1 (Dense)                 (None, 64)           4160        activation[0][0]
_________________________________________________________________
↪_______________
batch_normalization_1 (BatchNor (None, 64)           192         dense_1[0][0]
_________________________________________________________________
↪_______________
activation_1 (Activation)       (None, 64)           0           batch_normalization_
↪1[0][0]
_________________________________________________________________
↪_______________
concatenate (Concatenate)       (None, 2064)         0           encoder_input[0][0]
                                                                 activation_1[0][0]
_________________________________________________________________
↪_______________
dense_2 (Dense)                 (None, 128)          264320      concatenate[0][0]
_________________________________________________________________
↪_______________
batch_normalization_2 (BatchNor (None, 128)          384         dense_2[0][0]
_________________________________________________________________
↪_______________
activation_2 (Activation)       (None, 128)          0           batch_normalization_
↪2[0][0]
_________________________________________________________________
↪_______________
dropout (Dropout)               (None, 128)          0           activation_2[0][0]
_________________________________________________________________
↪_______________
dense_3 (Dense)                 (None, 64)           8256        dropout[0][0]
_________________________________________________________________
↪_______________
batch_normalization_3 (BatchNor (None, 64)           192         dense_3[0][0]
_________________________________________________________________
↪_______________
activation_3 (Activation)       (None, 64)           0           batch_normalization_
↪3[0][0]
_________________________________________________________________
↪_______________
dropout_1 (Dropout)             (None, 64)           0           activation_3[0][0]
_________________________________________________________________
↪_______________
```

(continues on next page)

(continued from previous page)

```
dense_4 (Dense)                 (None, 32)           2080        dropout_1[0][0]
↪_____________
batch_normalization_4 (BatchNor (None, 32)           96          dense_4[0][0]
↪_____________
tf.math.subtract (TFOpLambda)   (None, 2000)         0           encoder_input[0][0]
                                                                 decoder_mlp[0][0]
↪_____________
activation_4 (Activation)       (None, 32)           0           batch_normalization_
↪4[0][0]
↪_____________
tf.math.reduce_mean (TFOpLambda (1, 1)               0           tf.math.
↪subtract[0][0]
↪_____________
dropout_2 (Dropout)             (None, 32)           0           activation_4[0][0]
↪_____________
tf.math.subtract_1 (TFOpLambda) (None, 2000)         0           tf.math.
↪subtract[0][0]
                                                                 tf.math.reduce_
↪mean[0][0]
↪_____________
encoder_log_var (Dense)         (None, 16)           528         dropout_2[0][0]
↪_____________
encoder_mean (Dense)            (None, 16)           528         dropout_2[0][0]
↪_____________
tf.convert_to_tensor (TFOpLambd (None, 2000)         0           decoder_mlp[0][0]
↪_____________
tf.cast (TFOpLambda)            (None, 2000)         0           encoder_input[0][0]
↪_____________
tf.math.square (TFOpLambda)     (None, 2000)         0           tf.math.subtract_
↪1[0][0]
↪_____________
tf.__operators__.add_1 (TFOpLam (None, 16)           0           encoder_log_var[0][0]
↪_____________
tf.math.square_1 (TFOpLambda)   (None, 16)           0           encoder_mean[0][0]
↪_____________
tf.math.squared_difference (TFO (None, 2000)         0           tf.convert_to_
↪tensor[0][0]
                                                                 tf.cast[0][0]
↪_____________
tf.math.reduce_mean_1 (TFOpLamb ()                   0           tf.math.square[0][0]
↪_____________
```

(continues on next page)

(continued from previous page)

```
tf.math.subtract_2 (TFOpLambda) (None, 16)           0           tf.__operators__.add_
↪1[0][0]
                                                                 tf.math.square_
↪1[0][0]
__________________________________________________________________________________
↪_______________
tf.math.exp (TFOpLambda)        (None, 16)           0           encoder_log_var[0][0]
__________________________________________________________________________________
↪_______________
tf.math.reduce_mean_2 (TFOpLamb (None,)              0           tf.math.squared_
↪difference[0][0]
__________________________________________________________________________________
↪_______________
tf.math.truediv (TFOpLambda)    ()                   0           tf.math.reduce_mean_
↪1[0][0]
__________________________________________________________________________________
↪_______________
tf.math.log (TFOpLambda)        ()                   0           tf.math.reduce_mean_
↪1[0][0]
__________________________________________________________________________________
↪_______________
tf.math.subtract_3 (TFOpLambda) (None, 16)           0           tf.math.subtract_
↪2[0][0]
                                                                 tf.math.exp[0][0]
__________________________________________________________________________________
↪_______________
tf.math.multiply (TFOpLambda)   (None,)              0           tf.math.reduce_mean_
↪2[0][0]
                                                                 tf.math.truediv[0][0]
__________________________________________________________________________________
↪_______________
tf.math.multiply_1 (TFOpLambda) ()                   0           tf.math.log[0][0]
__________________________________________________________________________________
↪_______________
tf.math.reduce_sum (TFOpLambda) (None,)              0           tf.math.subtract_
↪3[0][0]
__________________________________________________________________________________
↪_______________
tf.__operators__.add (TFOpLambd (None,)              0           tf.math.
↪multiply[0][0]
                                                                 tf.math.multiply_
↪1[0][0]
__________________________________________________________________________________
↪_______________
tf.math.multiply_2 (TFOpLambda) (None,)              0           tf.math.reduce_
↪sum[0][0]
__________________________________________________________________________________
↪_______________
tf.math.multiply_3 (TFOpLambda) (None,)              0           tf.__operators__.
↪add[0][0]
__________________________________________________________________________________
↪_______________
tf.math.multiply_4 (TFOpLambda) (None,)              0           tf.math.multiply_
↪2[0][0]
__________________________________________________________________________________
↪_______________
tf.__operators__.add_2 (TFOpLam (None,)              0           tf.math.multiply_
↪3[0][0]
```

(continues on next page)

(continued from previous page)

```
                                                                    tf.math.multiply_
↪4[0][0]
__________________________________________________________________________________
↪______________
tf.math.reduce_mean_3 (TFOpLamb ()                   0           tf.__operators__.add_
↪2[0][0]
__________________________________________________________________________________
↪______________
add_loss (AddLoss)              ()                   0           tf.math.reduce_mean_
↪3[0][0]
==================================================================================================
Total params: 827,836
Trainable params: 826,080
Non-trainable params: 1,756
__________________________________________________________________________________
↪______________
Epoch 1/20
68/68 [==============================] - 5s 24ms/step - loss: 5699.5386
Epoch 2/20
68/68 [==============================] - 1s 22ms/step - loss: 5387.9050
Epoch 3/20
68/68 [==============================] - 2s 28ms/step - loss: 5354.2480
Epoch 4/20
68/68 [==============================] - 2s 23ms/step - loss: 5326.5728
Epoch 5/20
68/68 [==============================] - 2s 25ms/step - loss: 5316.6463
Epoch 6/20
68/68 [==============================] - 1s 22ms/step - loss: 5305.0356
Epoch 7/20
68/68 [==============================] - 1s 19ms/step - loss: 5304.1235
Epoch 8/20
68/68 [==============================] - 1s 19ms/step - loss: 5302.7018
Epoch 9/20
68/68 [==============================] - 1s 19ms/step - loss: 5290.4808
Epoch 10/20
68/68 [==============================] - 1s 19ms/step - loss: 5281.9169
Epoch 11/20
68/68 [==============================] - 1s 19ms/step - loss: 5289.3734
Epoch 12/20
68/68 [==============================] - 1s 20ms/step - loss: 5285.7788
Epoch 13/20
68/68 [==============================] - 1s 19ms/step - loss: 5285.3835
Epoch 14/20
68/68 [==============================] - 1s 18ms/step - loss: 5288.9461
Epoch 15/20
68/68 [==============================] - 1s 19ms/step - loss: 5277.2054
Epoch 16/20
68/68 [==============================] - 1s 20ms/step - loss: 5281.0600
Epoch 17/20
68/68 [==============================] - 1s 19ms/step - loss: 5281.3589
Epoch 18/20
68/68 [==============================] - 1s 20ms/step - loss: 5275.7597
Epoch 19/20
68/68 [==============================] - 1s 19ms/step - loss: 5276.1251
Epoch 20/20
68/68 [==============================] - 1s 20ms/step - loss: 5279.9665
WARNING:tensorflow:Found duplicated `Variable`s in Model's `weights`. This is usually␣
↪caused by `Variable`s being shared by Layers in the Model. These `Variable`s will␣
↪be treated as separate `Variable`s when the Model is restored. To avoid this,␣
↪please save with `save_format="tf"`.
```

(continues on next page)

(continued from previous page)

```
... storing '_batch' as categorical
... storing 'celltype' as categorical
... storing 'tech' as categorical
```

Cell type prediction

[5]:
```
atac=tfl.findNeighbors(adata_integrate)
```

store celltype in adata_integrate.obs[‘celltype’]

[6]:
```
adata_atac.obs['celltype'] = atac['celltype']
adata = adata_rna.concatenate(adata_atac)
adata_integrate.obs['celltype'] = adata.obs['celltype']
```

Loading result

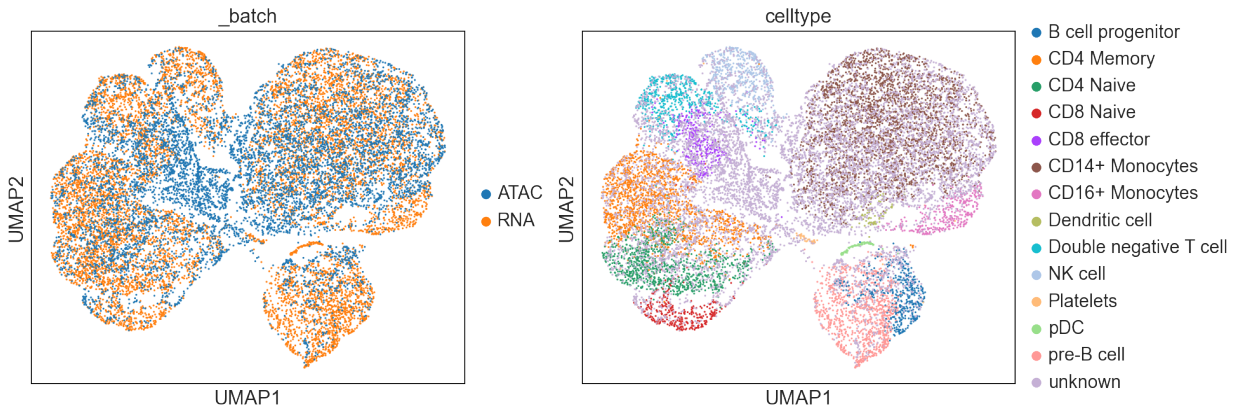
Loading result from h5ad file: The output.h5ad file of the trained result can be downloaded here

[7]:
```
adata_integrate = tl.read_sc_data('./atac_result/output_save.h5ad')
```

UMAP Visualization

[8]:
```
sc.pp.neighbors(adata_integrate, use_rep='X_vipcca')
sc.tl.umap(adata_integrate)

sc.set_figure_params(figsize=[5.5, 4.5])
sc.pl.umap(adata_integrate, color=['_batch', 'celltype'])
```



2.2 Indices and tables

- genindex
- modindex
- search

CHAPTER

THREE

VIPCCA API

3.1 API

3.1.1 Preprocessing

vipcca.tools.utils.**preprocessing**(*datasets*, *min_cells=1*, *min_genes=1*, *n_top_genes=2000*, *mt_ratio=0.8*, *lognorm=True*, *hvg=True*, *index_unique=None*)

Preprocess and merge data sets from different batches

Parameters

- **datasets** (*list, optional (default: None)*) – the list of anndata objects from different batches
- **min_cells** (*int, optional (default: 1)*) – Minimum number of counts required for a cell to pass filtering.
- **min_genes** (*int, optional (default: 1)*) – Minimum number of counts required for a gene to pass filtering.
- **n_top_genes** (*int, optional (default: 2000)*) – Number of highly-variable genes to keep.
- **mt_ratio** (*double, optional (default: 0.8)*) – Maximum proportion of mito genes for a cell to pass filtering.
- **lognorm** (*bool, optional (default: True)*) – If True, execute lognorm() function.
- **hvg** (*bool, optional (default: True)*) – If True, choose hypervariable genes for AnnData object.
- **index_unique** (*string, optional (default: None)*) – Make the index unique by joining the existing index names with the batch category, using index_unique='-', for instance. Provide None to keep existing indices.

Returns adata_norm

Return type AnnData

vipcca.tools.utils.**read_sc_data**(*input_file*, *fmt='h5ad'*, *backed=None*, *transpose=False*, *sparse=False*, *delimiter=' '*, *unique_name=True*, *batch_name=None*, *var_names='gene_symbols'*)

Read single cell dataset

Parameters

- **input_file** (*string*) – The path of the file to be read.
- **fmt** (*string, optional (default: 'h5ad')*) – The file type of the file to be read.
- **backed** (*Union[Literal['r', 'r+'], bool, None] (default: None)*) – If 'r', load AnnData in backed mode instead of fully loading it into memory (memory mode). If you want to modify backed attributes of the AnnData object, you need to choose 'r+'.
- **transpose** (*bool, optional (default: False)*) – Whether to transpose the read data.
- **sparse** (*bool, optional (default: False)*) – Whether the data in the dataset is stored in sparse matrix format.
- **delimiter** (*str, optional (default: ' ')*) – Delimiter that separates data within text file. If None, will split at arbitrary number of white spaces, which is different from enforcing splitting at single white space ' '.
- **unique_name** (*bool, optional (default: False)*) – If Ture, AnnData object execute var_names_make_unique() and obs_names_make_unique() functions.
- **batch_name** (*string, optional (default: None)*) – Batch name of current batch data
- **var_names** (*Literal['gene_symbols', 'gene_ids'] (default: 'gene_symbols')*) – The variables index when the file type is 'mtx'.

Returns adata

Return type AnnData

vipcca.tools.utils.**spatial_preprocessing**(*datasets*, *min_cells=1*, *min_genes=1*, *n_top_genes=2000*, *lognorm=True*, *hvg=True*)

Preprocess and merge two visium datasets from different batches

Parameters

- **datasets** (*list, optional (default: None)*) – The list of anndata objects from different batches
- **min_cells** (*int, optional (default: 1)*) – Minimum number of counts required for a cell to pass filtering.
- **min_genes** (*int, optional (default: 1)*) – Minimum number of counts required for a gene to pass filtering.
- **n_top_genes** (*int, optional (default: 2000)*) – Number of highly-variable genes to keep.
- **lognorm** (*bool, optional (default: True)*) – If True, execute lognorm() function.
- **hvg** (*bool, optional (default: True)*) – If True, choose hypervariable genes for AnnData object.

Returns adata

Return type AnnData

vipcca.tools.utils.**spatial_rna_preprocessing**(*adata_spatial*, *adata_rna*, *lognorm=True*, *hvg=True*, *n_top_genes=2000*)

Preprocess and merge visium dataset with scRNA-seq dataset.

Parameters

- **adata_spatial** (*AnnData*) – AnnData object of visium dataset.
- **adata_rna** (*AnnData*) – AnnData object of scRNA-seq dataset.
- **lognorm** (*bool, optional (default: True)*) – If True, execute lognorm() function.
- **hvg** (*bool, optional (default: True)*) – If True, choose hypervariable genes for AnnData object.
- **n_top_genes** (*int, optional (default: 2000)*) – Number of highly-variable genes to keep.

Returns adata

Return type AnnData

3.1.2 Plotting

vipcca.tools.plotting.**plotCorrelation**(*y*, *y_pred*, *save=True*, *result_path='./'*, *show=True*, *rnum=10000.0*, *lim=20*)

Plot correlation between original data and corrected data

Parameters

- **y** (*matrix or csr_matrix*) – The original data matrix.
- **y_pred** (*matrix or csr_matrix*) – The data matrix integrated by vipcca.
- **save** (*bool, optional (default: True)*) – If True, save the figure into result_path.
- **result_path** (*string, optional (default: './')*) – The path for saving the figure.
- **show** (*bool, optional (default: True)*) – If True, show the figure.
- **rnum** (*double, optional (default: 1e4)*) – The number of points you want to sample randomly in the matrix.
- **lim** (*int, optional (default: 20)*) – the right parameter of matplotlib.pyplot.xlim(left, right)

3.1.3 VIPCCA

class vipcca.model.vipcca.**VIPCCA**(*adata_all=None*, *patience_es=50*, *patience_lr=25*, *epochs=100*, *res_path=None*, *split_by='_batch'*, *method='lognorm'*, *hvg=True*, *batch_input_size=128*, *batch_input_size2=16*, *activation='softplus'*, *dropout_rate=0.01*, *hidden_layers=[128, 64, 32, 16]*, *lambda_regulizer=5.0*, *initializer='glorot_uniform'*, *l1_l2=(0.0, 0.0)*, *mode='CVAE'*, *model_file=None*, *save=True*)

Bases: object

Initialize VIPCCA object

Parameters

patience_es: int, optional (default: 50) number of epochs with no improvement after which training will be stopped.

patience_lr: int, optional (default: 25) number of epochs with no improvement after which learning rate will be reduced.

epochs: int, optional (default: 100) Number of epochs to train the model. An epoch is an iteration over the entire x and y data provided.

res_path: string, (default: None) Folder path to save model training results model.h5 and output data adata.h5ad.

split_by: string, optional (default: '_batch') the obsm_name of obsm used to distinguish different batches.

method: string, optional (default: 'lognorm') the normalization method for input data, one of {"qqnorm","count", other}.

batch_input_size: int, optional (default: 128) the length of the batch vector that concatenate with the input layer.

batch_input_size2: int, optional (default: 16) the length of the batch vector that concatenate with the latent layer.

activation: string, optional (default: "softplus") the activation function of hidden layers.

dropout_rate: double, optional (default: 0.01) the dropout rate of hidden layers.

hidden_layers: list, optional (default: [128,64,32,16]) Number of hidden layer neurons in the model

lambda_regulizer: double, optional (default: 5.0) The coefficient multiplied by KL_loss

initializer: string, optional (default: "glorot_uniform") Regularizer function applied to the kernel weights matrix.

l1_l2: tuple, optional (default: (0.0, 0.0)) [L1 regularization factor, L2 regularization factor].

mode: string, optional (default: 'CVAE') one of {"CVAE", "CVAE2", "CVAE3"}

model_file: string, optional (default: None) The file name of the trained model, the default is None

save: bool, optional (default: True) If true, save output adata file.

build()
build VIPCCA model

fit_integrate()
Train the constructed VIPCCA model, integrate the data with the trained model, and return the integrated anndata object

Returns adata produced by function self.conf.net.integrate(self.conf.adata_all, save=self.conf.save)

Return type AnnData

vipcca_preprocessing()
Generate the required random batch id for the VIPCCA model

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

V

INDEX