Holoclean Documentation

Release 0.1.0

Holoclean team

Apr 12, 2018

Contents

CHAPTER 1

Holoclean

Noisy and erroneous data is a major bottleneck in analytics. Data cleaning and repairing account for about 60% of the work of data scientists. To address this bottleneck, we recently introduced HoloClean, a semi-automated data repairing framework that relies on statistical learning and inference to repair errors in structured data. In HoloClean, we build upon the paradigm of weak supervision and demonstrate how to leverage diverse signals, including user-defined heuristic rules (such as generalized data integrity constraints) to repair erroneous data.

1.1 Holoclean and Session class

class holoclean.holoclean.**HoloClean**(***kwargs*)
Main entry point for HoloClean which creates a HoloClean Data Engine and initializes Spark.

class holoclean.holoclean.**Session**(*holo_env*, *name='session'*)
Session class controls the entire pipeline of HoloClean

add_denial_constraint(*dc*)
Adds denial constraints from string into self.Denial_constraints

Parameters **dc** – string in dc format

Returns string array of dc's

compare_to_truth(*truth_path*)
Compares our repaired set to the truth prints precision and recall

Parameters **truth_path** – path to clean version of dataset

detect_errors(*detector_list*)
Separates cells that violate DC's from those that don't

Parameters **detector_list** – List of error detectors

Returns clean dataframe

Returns don't know dataframe

load_clean_data(*file_path*)
Loads pre-defined clean cells from csv

Parameters **file_path** – path to file

Returns spark dataframe of clean cells

load_data(*file_path*)
Loads a dataset from file into the database

Parameters **file_path** – path to data file

Returns pyspark dataframe

load_denial_constraints(*file_path*)
Loads denial constraints from line-separated txt file

Parameters **file_path** – path to dc file

Returns string array of dc's

load_dirty_data(*file_path*)
Loads pre-defined dirty cells from csv

Parameters **file_path** – path to file

Returns spark dataframe of dirty cells

remove_denial_constraint(*index*)
Removing the denial constraint at index

Parameters **index** – index in list

Returns string array of dc's

repair()
Repairs the initial data includes pruning, featurization, and softmax

Returns repaired dataset

CHAPTER 2

Error detection

HoloClean learns to clean data by first splitting it into two categories *clean* and *dont_know* or *dk* for short. We've provided two kinds of error detectors, the *SqlDCErrorDetection* which uses Denial Constraints to make these splits and the *SqlnullErrorDetection* which labels all the cells that have null values as don't know cells. HoloClean also support custom or user-defined error detection methods by creating a new class that inherits from *ErrorDetection* and overrides the required methods.

2.1 ErrorDetector

class holoclean.errordetection.errordetector.**ErrorDetection**(*holo_obj*, *dataset*)

This class is an abstract class for general error_detection , it requires for every sub-class to implement the get_clean_cells and get_noisy_cells method

get_clean_cells()

This method creates a dataframe which has the information (index,attribute) for the clean_cells

:return dataframe for the clean_cells

get_noisy_cells()

This method creates a dataframe which has the information (index,attribute) for the dk_cells

:return dataframe for the dk_cell

2.2 SqlDCErrorDetection

class holoclean.errordetection.sql_dcerrordetector.**SqlDCErrorDetection**(*session*)

This class is a subclass of ErrorDetection class and will returns don't know cells and clean cells based on the denial constraints

get_clean_cells()

Returns a dataframe that consists of index of clean cells index, attribute

Returns spark dataframe

get_noisy_cells()

Returns a dataframe that consists of index of noisy cells index, attribute

Returns spark_dataframe

2.3 SqlnullErrorDetection

class holoclean.errordetection.sql_dcerrordetector.**SqlDCErrorDetection**(*session*)

This class is a subclass of ErrorDetection class and will returns don't know cells and clean cells based on the denial constraints

get_clean_cells()

Returns a dataframe that consists of index of clean cells index, attribute

Returns spark dataframe

get_noisy_cells()

Returns a dataframe that consists of index of noisy cells index, attribute

Returns spark_dataframe

CHAPTER 3

Featurization

Holoclean uses different signals to create the features for the model that it will use for the learning part. We've provided three kinds of signals: *SignalInit*, *SignalDC*, *SignalCooccur*. HoloClean also support custom or user-defined signal by creating a new class that inherits from *Featurizer* and overrides the required methods.

3.1 Featurizer

class holoclean.featurization.featurizer.**Featurizer**(*session*)

This class is an abstract class for general featurizer, it requires for every sub-class to implement the get_query method

get_query()

This method creates a string or strings of the query/queries that are used to create the Signal

:return a string or a list of strings of the query/queries that are used to create the Signal

3.2 SignalInit

class holoclean.featurization.initfeaturizer.**SignalInit**(*session*)

This class is a subclass of Featurizer class and will return the query which represent the Initial Signal for the clean and don't know cells

get_query(*clean=1*)

Creates a string for the query that it is used to create the Initial Signal

Parameters **clean** – shows if create the feature table for the clean or

the don't know cells

:return a list of length 1 with a string with the query for this feature

3.3 SignalDC

class holoclean.featurization.dcfeaturizer.**SignalDC**(*denial_constraints*, *session*)

This class is a subclass of the Featurizer class and will return a list of queries which represent the DC Signal for the clean and don't know cells

get_query(*clean=1*, *dcquery_prod=None*)

Creates a list of strings for the queries that are used to create the DC Signals

Parameters **clean** – shows if we create the feature table for the clean or the

dk cells :param dcquery_prod: a thread that we will produce the final queries

:return a list of strings for the queries for this feature

3.4 SignalCooccur

class holoclean.featurization.cooccurrencefeaturizer.**SignalCooccur**(*session*)

This class is a subclass of Featurizer class for the co-occur signal and will fill the tensor

get_query(*clean=1*)

Adding co-occur feature

Parameters **clean** – shows if create the feature table for the clean or the dk cells

:return list

insert_to_tensor(*tensor*, *clean*)

Inserting co-occur data into tensor

Parameters

- **tensor** – tensor object
- **clean** – Nat value that identifies if we are calculating feature

value for training data (clean cells) or testing data

Returns None

CHAPTER 4

Repairing

The clean cells are used as training examples to learn the parameters (weights) of a softmax regression model. Once those weights are defined, we use this model to perform inference on the "don't-know" cells and insert the most likely value for each cell.

4.1 Softmax

class holoclean.learning.softmax.**SoftMax**(*session*, *X_training*)

build_model(*featurizers*, *input_dim_non_dc*, *input_dim_dc*, *output_dim*, *tie_init=True*, *tie_DC=True*)

Initializes the logreg part of our model

Parameters

- **input_dim_non_dc** – number of init + cooccur features
- **featurizers** – list of featurizers
- **input_dim_dc** – number of dc features
- **output_dim** – number of classes
- **tie_init** – boolean to decide weight tying for init features
- **tie_DC** – boolean to decide weight tying for dc features

Returns newly created LogReg model

log_weights()

Writes weights in the logger

Returns Null

logreg(*featurizers*)

Trains our model on clean cells and predicts vals for clean cells

Returns predictions

predict(*model*, *x_val*, *mask=None*)
Runs our model on the test set

Parameters

- **model** – trained logreg model
- **x_val** – test x tensor
- **mask** – masking tensor to restrict domain

Returns predicted classes with probabilities

save_prediction(*Y*)
Stores our predicted values in the database

Parameters **Y** – tensor with probability for each class

Returns Null

setupMask(*clean=1*, *N=1*, *L=1*)
Initializes a masking tensor for ignoring impossible classes

Parameters

- **clean** – 1 if clean cells, 0 if don't-know
- **N** – number of examples
- **L** – number of classes

Returns masking tensor

setuptrainingX(*sparse=0*)
Initializes an X tensor of features for training

Parameters **sparse** – 0 if dense tensor, 1 if sparse

Returns x tensor of features

train(*model*, *loss*, *optimizer*, *x_val*, *y_val*, *mask=None*)
Trains our model on the clean cells

Parameters

- **model** – logistic regression model
- **loss** – loss function used for evaluating performance
- **optimizer** – optimizer for our neural net
- **x_val** – x tensor - features
- **y_val** – y tensor - output for comparison
- **mask** – masking tensor

Returns cost of traininng

Python Module Index

h

Index

R

S

T