MSCO - Multi-Stage Classification Optimization

Generated by Doxygen 1.8.13

Chapter 1

README

MSCO: Multi-Stage Classification Optimization

MSCO is a project in development aiming to provide tools which help users construct and optimize multi-stage classification schemes.

Please see `docs` directory for example workflows, method comparisons, and more background info.

More advanced methods involving genetic algorithms, which are outside the scope of this project, are also being developed with Errin Fulp (http://haminh16.sites.wfu.edu/GECCO2020.pdf)

1.1 MSCO Namespace Reference

Functions

- def check_partition_criteria (partition, phase_feature_mins=None, phase_feature_reqs=None)
- def partition_dataset (X, partition)
- def randomize_partition (partition, n, max_stage_ct, phase_feature_reqs=[ ], phase_feature_mins=[ ], p_add=0.05, p_mod=0.25)
- def make_stages (clf, X, y, partition)
- def process_input (parted_X, trained_models, x, prob_thresh=.75)
- def staged_classify (clf, X, y, partition, feature_costs, prob_thresh=.75, train_percent=.8, all_costs=[ ])
- def jenks_stages (clf, X, y, feature_costs, max_k, min_max_norm=True, prob_thresh=.9, train_percent=.75)
- def n_stages (clf, X, y, feature_costs, min_max_norm=True, prob_thresh=.9, train_percent=.75)
- def beam (clf, X, y, partition, feature_costs, pop_size=50, max_iter=10, beam_percent=0.1, stage_inc_max=3)
- def deterministic_assn (clf, X, y, K, seed, feature_costs, max_iter=3)
- def stochastic_assn (N, K, feature_costs, feature_benefits, init_choices=[ ])
- def pm_euclidean (scores, acc_coef=1, conc_coef=1, inv_time_coef=1)

1.1.1 Detailed Description

```
MSCO provides tools for constructing and optimizing multi-stage classifiers
from ordered feature set partitions
```

1.1.2 Function Documentation

1.1.2.1 beam()

```
def MSCO.beam (
              clf,
              X,
              y,
              partition,
              feature_costs,
              pop_size = 50,
              max_iter = 10,
              beam_percent = 0.1,
              stage_inc_max = 3 )
```

```
perform a simple beam search on the solution space of multi-stage designs

Args:
    clf: scikit-learn classifier object
    X: pandas df of records for training/testing
    y: labels for records in `X`
    partition: a list containing integer elements with the value at
      index i denoting the stage assignment of feature i
    feature_costs: list of values corresponding to the runtime costs
      of features in `X`.
    pop_size: denotes the size of generated population from which to
      select beam population
    max_iter: maximum number of generations (populations)

Returns:
    a list containing the best performing partition and its performance
```

1.1.2.2 check_partition_criteria()

```
def MSCO.check_partition_criteria (
              partition,
              phase_feature_mins = None,
              phase_feature_reqs = None )
```

```
check if `partition` satisfies given criteria

Args:
    partition: a list containing integer elements with the value at
      index i denoting the stage assignment of feature i
    phase_feature_mins: a list containing integer elements with
      the value at index i denoting the minimum number of features
      that must be assigned to stage i.
    phase_feature_reqs: a list containing integer elements with
      the value at index i denoting the required stage assignment
      for feature i. if no required stage assignment is desired
      for feature i, then phase_feature_reqs[i] should be set to
      -1.

Returns:
    True if criteria is satisfied
```

1.1.2.3 deterministic_assn()

```
def MSCO.deterministic_assn (
              clf,
              X,
              y,
              K,
              seed,
              feature_costs,
              max_iter = 3 )
```

```
deterministic sequential feature assignment method described in 3.6.1

Args:
    clf: sklearn classifier object
    X: pandas dataframe containing records for training/testing
    y: array-like object containing labels for corresponding records
    feature_costs: array-like object containing float values corresponding to
        "costs" of features as defined in MSCO/docs/hamilton_thesis.pdf
    seed: beginning partition to modify iteratively with deterministic method
    max_iter: number of iterations through all feature assignments, modifying
        each feature assignment on each iteration

Returns:
    solution obtained after `max_iter` runs through list of feature assignments.
```

1.1.2.4 jenks_stages()

```
def MSCO.jenks_stages (
              clf,
              X,
              y,
              feature_costs,
              max_k,
              min_max_norm = True,
              prob_thresh = .9,
              train_percent = .75 )
```

```
use jenks natural breaks optiimization on costs to create multi-stage models

Args:
    clf: sklearn classifier object
    X: pandas dataframe containing records for training/testing
    y: array-like object containing labels for corresponding records
    feature_costs: array-like object containing float values corresponding to
        "costs" of features as defined in MSCO/docs/hamilton_thesis.pdf
    max_k: maximum number of allowed stages to consider
    min_max_norm: if True, scale feature costs to [0,1]

Returns:
    a list of solutions generated from jenks using varying K in [2, max_k]
        along with their respective performance
```

1.1.2.5 make_stages()

```
def MSCO.make_stages (
              clf,
              X,
              y,
              partition )

create a multi-stage classification scheme according to `partition`

Args:
    clf: sklearn classifier object
`       X: pandas dataframe containing training and test records
    y: array-like object containing labels for corresponding records
    partition: array-like object containing stage assignments for
        features. partition[i] is the stage assignment for feature
        i.

Returns:
    partitioned dataframe AND a list of stages/subclassifiers
```

1.1.2.6 n_stages()

```
def MSCO.n_stages (
              clf,
              X,
              y,
              feature_costs,
              min_max_norm = True,
              prob_thresh = .9,
              train_percent = .75 )

create an N-stage model of increasing feature cost

Args:
    clf: sklearn classifier object
    X: pandas dataframe containing records for training/testing
    y: array-like object containing labels for corresponding records
    feature_costs: array-like object containing float values corresponding to
        "costs" of features as defined in MSCO/docs/hamilton_thesis.pdf

Returns:
    n-stage solution and correponding performance score
```

1.1.2.7 partition_dataset()

```
def MSCO.partition_dataset (
              X,
              partition )

partition dataframe `X` according to `partition`

Args:
    X: a pandas dataframe containing records for training/testing
    partition: a list containing integer elements with the value at
      index i denoting the stage assignment of feature i

Returns:
    a list of pandas dataframes representing the partitioned dataset
```

1.1.2.8 pm_euclidean()

```
def MSCO.pm_euclidean (
              scores,
              acc_coef = 1,
              conc_coef = 1,
              inv_time_coef = 1 )
```

```
default performance metric based on euclidean distance from origin

Args:
    scores: scores dict returned from staged_classify()
    acc_coef: accuracy multiplier
    conc_coef: conclusiveness multiplier
    inv_time_coef: inverse time multiplier

Returns:
    3-dimensional euclidean distance from origin (0,0,0) to
    (scaled_acc, scaled_conc, scaled_inv_time)
```

1.1.2.9 process_input()

```
def MSCO.process_input (
              parted_X,
              trained_models,
              x,
              prob_thresh = .75 )
```

```
evaluate inputs with multi-stage model

Args:
    parted_X: returned from make_stages()[0], this is a subset
      of X containing only features included in stage i
    trained_models: submodels (stages) trained on parted_X
    x: the input to be processed. must have all features in
      initial dataset (X)
    prob_thresh: probability threshold for conclusiveness

Returns:
    an index associated with the respective label/class.
    if processing is not conclusive, -1 is returned
```

1.1.2.10 randomize_partition()

```
def MSCO.randomize_partition (
              partition,
              n,
              max_stage_ct,
              phase_feature_reqs = [],
              phase_feature_mins = [],
              p_add = 0.05,
              p_mod = 0.25 )
```

```
generate random population of partitions from `partition`

Args:
    partition: a list containing integer elements with the value at
      index i denoting the stage assignment of feature i
    n: number of new partitions to generate
    p_add: probability of adding a new stage
    p_mod: probability of modifying the existing stage assignment
    max_stage_ct: stage assignment values will not be set higher than this

Raises:
    ValueError: if `partition` does not satisfy criteria

Returns:
    a list of lists representing partitions of the initial feature set.
```

1.1.2.11 staged_classify()

```
def MSCO.staged_classify (
              clf,
              X,
              y,
              partition,
              feature_costs,
              prob_thresh = .75,
              train_percent = .8,
              all_costs = [] )

evaluate the accuracy/efficiency of a multi-stage model given by `partition`

Args:
    clf: scikit-learn classifier object
    X: pandas df of records for training/testing
    y: labels for records in `X`
    partition: a list containing integer elements with the value at
      index i denoting the stage assignment of feature i
    feature_costs: list of values corresponding to the runtime costs
      of features in `X`.
    prob_thresh: the minimum confidence in prediction that a stage
      in the model must have for a decision to be deemed conclusive
    train_percent: this value denotes the percentage of records that
      will be dedicated to training. (1 - `train-percent`) records
      will be used for testing

Returns:
    a dictionary related to the performance of the model:
      'correct': number of correctly classified records
      'incorrect': number of incorrectly classified records
      'inconclusive': number of records determined 'inconclusive'
      'runtime': total runtime of model according to values in `feature_costs`
```

1.1.2.12 stochastic_assn()

```
def MSCO.stochastic_assn (
              N,
              K,
              feature_costs,
              feature_benefits,
              init_choices = [] )
```

```
stochastic sequential feature assignment method described in 3.6.2

Args:
    clf: sklearn classifier object
    X: pandas dataframe containing records for training/testing
    y: array-like object containing labels for corresponding records
    feature_costs: array-like object containing float values corresponding to
        "costs" of features as defined in MSCO/docs/hamilton_thesis.pdf
    init_choices: array-like object of length K containing initial features
        to assign to stages 0...K-1

Returns:
    solution and performance obtained
```