scikit-weak

Release 0.1.0

Andrea Campagner

Jun 30, 2022

CONTENTS


scikit
WEAK

CHAPTER

ONE

WELCOME

Welcome to the documentation of scikit-weak, a library for weakly supervised learning inspired by scikit-learn.

scikit-weak can be installed via:

```
pip install scikit-weak
```

Alternatively, you can download the source from:

GitHub

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Classification

The classification module features several state-of-the-art classification algorithms for weakly supervised learning.

2.1.1 Table of Contents

GRM

The grm module contains classes to perform weakly supervised classification based on generalized risk minimization, adopting an optimization-based approach.

Table of Contents

GRMLinearClassifier

class scikit_weak.classification.**GRMLinearClassifier**(*loss='logistic'*, *max_epochs=100*, *optimizer='sgd'*, *random_state=None*, *regularizer=None*, *l1=0.01*, *l2=0.01*, *batch_size=32*)

A class to perform classification for weakly supervised data, based on the Generalized Risk Minimization Paradigm applied to linear classifiers (either, logistic regression or linear SVM). The y input to the fit method should be given as an iterable of DiscreteWeakLabel

Parameters

- **loss** (*str or callable, default 'logistic'*) – The loss function to optimize
- **max_epochs** (*int, default 100*) – The number of epochs to train the model
- **optimizer** (*str or callable, default 'sgd'*) – The optimizer algorithm
- **regularizer** (*str or callable, default None*) – The type of regularization to apply
- **l1** (*float, default 0.01*) – The regularization coefficient for l1 regularization, used only if regularizer='l1'
- **l2** (*float, default 0.01*) – The regularization coefficient for l2 regularization, used only if regularizer='l2'
- **batch_size** (*int, default 32*) – Size of the mini-batches

Variables

- **regularization** – The regularization method instance
- **model** (*tf.keras.Model*) – The fitted linear model
- **loss_funct** (*callable*) – The loss function callable
- **__n_classes** (*int*) – The number of unique classes in y
- **__classes** (*list of int*) – The unique classes in y

fit(*X*, *y*)

Fit the GRMLinearClassifier model

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

References

[1] Hüllermeier, E. (2014).
Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. International Journal of Approximate Reasoning, 55(7), 1519-1534. https://doi.org/10.1016/j.ijar.2013.09.003

Ensemble

The ensemble module contains classes for weakly supervised classification based on ensemble methods.

Table of Contents

RRLClassifier

class scikit_weak.classification.**RRLClassifier**(*estimator=ExtraTreeClassifier()*, *n_estimators=100*, *resample=False*, *random_state=None*)

A class to perform classification for weakly supervised data, based on the RRL algorithm [1]. The y input to the fit method should be given as an iterable of DiscreteWeakLabel

Parameters

- **estimator** (*estimator class, default=ExtraTreeClassifier*) – Base estimator objects to be fitted. Should support predict and predict_proba
- **n_estimators** (*int, default=100*) – The number of trees to be fitted
- **resample** (*bool, default=False*) – Whether to perform bootstrapping or not
- **random_state** (*int, default=None*) – Random seed

Variables

- **classifiers** (*list of estimators*) – The collection of fitted estimators
- **__ys** (*list of ndarrays*) – The collection of sampled target labels. Each ndarray in ys has the same shape as y

- **__Xs** (*list of ndarrays*) – The collection of bootstrapped datasets. Each ndarray in Xs has the same shape as X. If resample=False, then Xs is empty.
- **__n_classes** (*int*) – The number of unique classes in y
- **__classes** (*list of int*) – The unique classes in y

fit(*X*, *y*)

Fit the RRLClassifier model

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

References

[1] Campagner, A., Ciucci, D., Svensson, C. M., Figge, M. T., & Cabitza, F. (2021).
Ground truthing from multi-rater labeling with three-way decision and possibility theory. Information Sciences, 545, 771-790. https://doi.org/10.1016/j.ins.2020.09.049

Neighbors

WeaklySupervisedKNeighborsClassifier

class scikit_weak.classification.**WeaklySupervisedKNeighborsClassifier**(*k=3*, *metric='minkowski'*)

A class to perform classification for weakly supervised data, based on k-nearest neighbors. The y input to the fit method should be given as an iterable of DiscreteWeakLabel

Parameters

- **k** (*int, default=3*) – The number of neighbors
- **metric** (*str or callable, default 'minkowski'*) – The metric for neighbors queries

Variables

- **__n_classes** (*int*) – The number of unique classes in y
- **__classes** (*list of int*) – The unique classes in y

fit(*X*, *y*)

Fit the WeaklySupervisedKNeighborsClassifier model

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

WeaklySupervisedRadiusClassifier

class scikit_weak.classification.**WeaklySupervisedRadiusClassifier**(*radius=1.0*, *metric='minkowski'*)

A class to perform classification for weakly supervised data, based on radius neighbors. The y input to the fit method should be given as an iterable of DiscreteWeakLabel

Parameters

- **radius** (*float, default=1.0*) – The size of the radius
- **metric** (*str or callable, default 'minkowski'*) – The metric for neighbors queries

Variables

- **y** (*ndarray*) – A copy of the input y
- **n_classes** (*int*) – The number of unique classes in y

fit(*X*, *y*)

Fit the WeaklySupervisedRadiusClassifier model

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

Pseudo Labels

The pseudo_labels module contains classes to perform weakly supervised classification based on the pseudo labeling approach, where labels generated by the classifier are iteratively incorporated in the training process.

CSSLClassifier

class scikit_weak.classification.**CSSLClassifier**

A credal self-supervised learning classifiers as described in "Credal Self-Supervised Learning" by Julian Lienen and Eyke Huellermeier, NeurIPS 2021. The model iteratively refines its beliefs about the true conditional class probability distribution for weakly labeled instances by maintaining credal sets induced by possibility distributions.

Parameters

- **estimator** (*estimator class, default=LabelRelaxationNNClassifier*) – Base estimator objects to be fitted. Should support predict and predict_proba
- **n_iterations** (*int, default=10*) – The number of iterations for fitting
- **random_state** (*int, default=None*) – Random seed
- **p_data** (*np.ndarray, default=None*) – Optional prior probability distribution of the class frequencies
- **p_hist_buffer_size** (*int, default=64*) – Buffer size of the model prediction history

Variables

- **estimator** (*estimator*) – The last fitted estimator
- **_n_classes** (*int*) – The number of unique classes in y

fit(*X*, *y*)

y is list of discrete fuzzy labels

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

References

[1] Lienen, J., Hüllermeier, E. (2021).
Credal Self-Supervised Learning. Advances in Neural Information Processing Systems, 34.

PseudoLabelsClassifier

class scikit_weak.classification.**PseudoLabelsClassifier**(*estimator=LogisticRegression()*, *n_iterations=10*, *n_restarts=5*, *threshold=0.5*, *random_state=None*)

A class to perform classification for weakly supervised data, based on the pseudo-labels strategy. The y input to the fit method should be given as an iterable of GenericWeakLabel

Parameters

- **estimator** (*estimator class, default=LogisticRegression*) – Base estimator objects to be fitted. Should support predict and predict_proba
- **n_restarts** (*int, default = 5*) – The number of restarts
- **n_iterations** (*int, default=10*) – The number of iterations for fitting
- **threshold** (*float, default=0.5*) – The threshold for pseudo-label selection
- **random_state** (*int, default=None*) – Random seed

Variables

- **estimator** (*estimator*) – The last fitted estimator
- **__n_classes** (*int*) – The number of unique classes in y
- **__classes** (*list of int*) – The unique classes in y

fit(*X*, *y*)

Fit the PseudoLabelsClassifier model

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

Label Relaxation

The relaxation module contains classes to perform label relaxation-based classification.

Table of Contents

LabelRelaxationNNClassifier

class scikit_weak.classification.**LabelRelaxationNNClassifier**(*lr_alpha: float = 0.1*, *hidden_layer_sizes: tuple = (100,)*, *activation: str = 'relu'*, *l2_penalty: float = 0.0001*, *learning_rate: float = 0.001*, *momentum: float = 0.0*, *epochs: int = 100*, *batch_size: Optional[int] = None*, *provide_alphas: bool = False*, *n_classes: Optional[int] = None*)

Simple MLP-based classifier using the label relaxation loss as optimization criterion.

Parameters

- **lr_alpha** (*float, default=0.1*) – Imprecisiation degree alpha of the label relaxation loss
- **hidden_layer_sizes** (*tuple, default=(100,)*) – Tuple consisting of the individual hidden layer sizes used by the underlying NN model
- **activation** (*str, default 'relu'*) – Activation function applied to the hidden layers' activations
- **l2_penalty** (*float, default=1e-4*) – L2 norm regularization parameter applied to all neural network layers
- **learning_rate** (*float, default=1e-3*) – Learning rate used by the SGD optimizer
- **momentum** (*float, default=0.0*) – Momentum used by the SGD optimizer
- **epochs** (*int, default=100*) – Number of training epochs
- **batch_size** (*int, default=None*) – Batch size for training
- **provide_alphas** (*bool, default=False*) – Indicator whether we consider tuples as targets consisting of the classes and their imprecisiation
- **n_classes** (*int default=None*) – Number of classes in case we want to be certain about the dimensionality of the one-hot encoding

fit(*X*, *y*)

Fits the label relaxation model. The targets y are one-hot encoded in case a simple list is provided.

References

[1] Lienen, J., Hüllermeier, E. (2021).
From label smoothing to label relaxation. Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI.

2.2 Feature Selection

The feature_selection module includes classes to perform feature selection and dimensionality reduction.

2.2.1 Table of Contents

DELIN

class scikit_weak.feature_selection.**DELIN**(*k=3*, *d=2*, *n_iters=10*)

A class to perform classification and dimensionality reduction for weakly supervised data, based on the DELIN algorithm [1]. The original method has been slighlty modified by using SVD in the computation of the inverse, so as to avoid issues when the data matrix is singular. The y input to the fit method should be given as an iterable of DiscreteWeakLabel

Parameters

- **k** (*int, default=3*) – The number of neighbors
- **d** – The number of dimensions to be kept after reduction. If int, then the exact number of features to be kept.

If float, the percentage of the total number of dimensions. :type d: int or float, default = 2

Parameters
iters (*int, default = 10*) – The number of iterations to be performed

Variables

- **y** (*ndarray*) – If y is in prob format, then target is a copy of y. Otherwise it is y in prob format
- **clf** (*WeaklyKNeighborsClassifier object*) – A WeaklyKNeighborsClassifier classifier to be used during fitting of the algorithm
- **vr** (*ndarray*) – A square ndarray with the same dim as X.shape[1]. Used to perform dimensionality reduction
- **n_classes** (*int*) – The number of unique classes in y
- **classes** (*ndarray*) – The unique classes in y

fit(*X*, *y*)

Fit the DELIN model

fit_transform(*X*, *y*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

X
[array-like of shape (n_samples, n_features)] Input samples.

y
[array-like of shape (n_samples,) or (n_samples, n_outputs), default=None] Target values (None for unsupervised transformations).

****fit_params**
[dict] Additional fit parameters.

X_new
[ndarray array of shape (n_samples, n_features_new)] Transformed array.

predict(*X*)

Returns predictions for the given X

predict_proba(*X*)

Returns probability distributions for the given X

References

[1] Wu, J. H., & Zhang, M. L. (2019).
Disambiguation enabled linear discriminant analysis for partial label dimensionality reduction. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19), 416-424. https://doi.org/10.1145/3292500.3330901

GeneticRoughSetSelector

class scikit_weak.feature_selection.**GeneticRoughSetSelector**(*epsilon=0.0*, *method='conservative'*, *discrete=False*, *l=0.5*, *tournament_size=0.1*, *p_mutate=0.1*, *metric='minkowski'*, *neighborhood='nearest'*, *n_neighbors=3*, *radius=1.0*, *random_state=None*, *population_size=100*, *n_iters=100*)

A class to perform Rough Set-based feature selection, by searching for reducts, using Genetic Algorithms. The y input to the fit method should be given as an iterable of DiscreteWeakLabel. Supports both discrete (using Pawlak Rough Sets) and continuous (using Neighborhood Rough Sets) datasets.

Parameters

- **epsilon** (*float, default=0.0*) – The approximation factor. Should be a number between 0.0 and 1.0 (excluded)
- **method** (*{'lambda', 'conservative', 'dominance'}, default='conservative'*) – The method used to compute the fitness. If 'lambda', then the algorithm solves a single objective optimization problem If 'conservative' or 'dominance' solves a multiple objective optimization problem: in particular, if 'conservative' a 2-objectives problem and if 'dominance' (n+1)-objectives problem, where n is the number of instances
- **discrete** (*bool, default=True*) – Whether the input X is discrete or not. If discrete=True then use equivalence-based (i.e. Pawlak) Rough Sets. If discrete=False use neighborhood-based Rough Sets.
- **l** (*float in [0,1], default=0.5*) – Lambda interpolation factor. Only used if method='lambda'

- **tournament_size** (*float in [0,1], default=0.1*) – Proportion of population to select from in tournament selection.
- **p_mutate** (*float in [0,1], default=0.1*) – Proability of point mutation
- **metric** (*string or function, default='minkowski'*) – Metric to be used with neighborhood-based Rough Sets. Only used if discrete=False. If discrete=True, then metric=”hamming”
- **neighborhood** (*{'delta', 'nearest'}, default='nearest'*) – Type of neighborhood-based Rough Sets to be used. If neighborhood=’delta’, then use delta-neighborhood Rough Sets: all neigbhors with distance <= radius are selected. If neighborhood=’nearest’, then use k-nearest-neighbors Rough Sets: only the k nearest neighbors are selected. Only used if discrete=False
- **n_neighbors** (*int, default=3*) – Number of nearest neighbors to select. Only used if discrete=False and neighborhood=’nearest’
- **radius** (*float, default=1.0*) – Radius to select neighbors. Only used if discrete=False and neighborhood=’delta’
- **random_state** (*int, default=None*) – Randomization seed. Used only if search_strategy=’approximate’
- **population_size** (*int, default=100*) – Size of the population for the Genetic Algorithm
- **n_iters** (*int, default=100*) – Number of generations for the Genetic Algorithm

Variables

- **n_classes** (*int*) – The number of unique classes in y
- **best_features** (*ndarray*) – The unique most fit feature sets.
- **best_targets** – The disambiguated targets corresponing to the most fit feature sets. Can be used for transductive learning or training a downstream model.

fit(*X*, *y*)

Fit the GeneticRoughSetSelector model

fit_transform(*X*, *y*)

Fit and then transform data

transform(*X*, *y=None*)

Transform the data (only X, y is ignored) selecting a reduct at random

RoughSetSelector

class scikit_weak.feature_selection.**RoughSetSelector**(*search_strategy='approximate'*, *epsilon=0.0*, *n_iters=100*, *method='conservative'*, *l=0.5*, *discrete=True*, *metric='minkowski'*, *neighborhood='nearest'*, *n_neighbors=3*, *radius=1.0*, *random_state=None*)

A class to perform Feature Selection based on Rough Sets by searching for reducts [1]. The y input to the fit method should be given as an iterable of DiscreteWeakLabel. Supports both discrete (using Pawlak Rough Sets) and continuous (using Neighborhood Rough Sets) datasets.

Parameters

- **search_strategy** (*{'approximate', 'brute'}, default='approximate'*) – The search strategy to be used. 'approximate' is similar to RFE, having complexity O(n^2). 'brute' is a brute-force search strategy, all possible combinations of features are evaluated, with complexity O(2^n)
- **epsilon** (*float, default=0.0*) – The approximation factor. Should be a number between 0.0 and 1.0 (excluded)
- **n_iters** (*int, default=100*) – Number of iterations to be used when search_strategy='approximate'. Not used if search_strategy='brute'
- **method** (*{'lambda', 'conservative'}, default='conservative'*) – The method used to compute the fitness. If 'lambda', then the algorithm solves a single objective optimization problem If 'conservative' solve a 2-objectives problem
- **l** (*float in [0,1], default=0.5*) – Lambda interpolation factor. Only used if method='lambda'
- **discrete** (*bool, default=True*) – Whether the input X is discrete or not. If discrete=True then use equivalence-based (i.e. Pawlak) Rough Sets. If discrete=False use neighborhood-based Rough Sets.
- **metric** (*string or function, default='minkowski'*) – Metric to be used with neighborhood-based Rough Sets. Only used if discrete=False. If discrete=True, then metric="hamming"
- **neighborhood** (*{'delta', 'nearest'}, default='nearest'*) – Type of neighborhood-based Rough Sets to be used. If neighborhood='delta', then use delta-neighborhood Rough Sets: all neigbhors with distance <= radius are selected. If neighborhood='nearest', then use k-nearest-neighbors Rough Sets: only the k nearest neighbors are selected. Only used if discrete=False
- **n_neighbors** (*int, default=3*) – Number of nearest neighbors to select. Only used if discrete=False and neighborhood='nearest'
- **radius** (*float, default=1.0*) – Radius to select neighbors. Only used if discrete=False and neighborhood='delta'
- **random_state** (*int, default=None*) – Randomization seed. Used only if search_strategy='approximate'

Variables

- **n_classes** (*int*) – The number of unique classes in y
- **reducts** (*list*) – The list of minimal reducts. If search_strategy='approximate', reducts always contains at most a single set of features for each membership degree. If search_strategy='brute', reducts contains the list of all minimal reducts.
- **reducts_poss** – The list of membership values of the minimal reducts.

fit(*X*, *y*)

Fit the RoughSetSelector model

fit_transform(*X*, *y*)

Fit and then transform data

transform(*X*, *y=None*)

Transform the data (only X, y is ignored) selecting a reduct at random

References

[1] Campagner, A., Ciucci, D., Hüllermeier, E. (2021).
Rough set-based feature selection for weakly labeled data. International Journal of Approximate Reasoning, 136, 150-167. https://doi.org/10.1016/j.ijar.2021.06.005.

[2] Campagner, A., Ciucci, D. (2021)
Feature Selection and Disambiguation in Learning from Fuzzy Labels Using Rough Sets. International Joint Conference on Rough Sets, LNCS 12872, 164-179. https://doi.org/10.1007/978-3-030-87334-9_14

[3] Campagner, A., Ciucci, D., & Hüllermeier, E. (2020).
Feature Reduction in Superset Learning Using Rough Sets and Evidence Theory. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, CCIS 1237, 471-484. https://doi.org/10.1007/978-3-030-50146-4_35

2.3 Utils

The utils module contain utility functions and classes for data pre-processing.

2.3.1 Table of Contents

Smoothers

The smoothers module contains classes to transform standard supervised datasets into weakly supervised data.

Table of Contents

DiscreteEstimatorSmoother

class scikit_weak.utils.**DiscreteEstimatorSmoother**(*estimator*, *type='set'*, *epsilon=1.0*)

A class to transform a supervised learning problem into a weakly supervised one, based on the output of a classifier. It currently supports transformation to superset and fuzzy label learning. Note that DiscreteEstimatorSmoother does not implement the transform method: therefore, usage should be based on calling fit_transform.

Parameters

- **estimator** (*estimator class*) – Base estimator objects to be fitted. Should support predict_proba
- **type** – Type of weakly supervised labels to transform into
- **epsilon** (*float, default=1.0*) – Parameter to select the minimum allowed label degree. Only used when type == 'set'. Should be between 0 and 1

Variables

n_classes (*int*) – The number of unique classes in y

fit_transform(*X*, *y*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

X
[array-like of shape (n_samples, n_features)] Input samples.

y
[array-like of shape (n_samples,) or (n_samples, n_outputs), default=None] Target values (None for unsupervised transformations).

****fit_params**
[dict] Additional fit parameters.

X_new
[ndarray array of shape (n_samples, n_features_new)] Transformed array.

DiscreteRandomSmoother

class scikit_weak.utils.**DiscreteRandomSmoother**(*p_err=0.1*, *p_incl=1.0*, *prob_ranges=None*, *type='set'*, *samples=100*, *epsilon=0.0*)

A class to transform a supervised learning problem into a weakly supervised one, based on random sampling. It currently supports transformation to superset and fuzzy label learning. Note that DiscreteRandomSmoother does not implement the transform method: therefore, usage should be based on calling fit_transform.

Parameters

- **p_err** (*float, default=0.1*) – The probability to include any single wrong label in a sample draw. Should be between 0 and 1.
- **p_incl** (*float, default=1.0*) – The probability to include the correct label in a sample draw. Should be between 0 and 1.
- **prob_ranges** (*enumerable of float, default=None*) – Array of possible membership degrees to be sampled. If not None, overrides both p_err and p_incl
- **type** (*{'set', 'fuzzy'}, default=set*) – Type of weakly supervised labels to transform into
- **samples** (*int, default=100*) – Number of samples to be generated
- **epsilon** (*float, default=1.0*) – Parameter to select the minimum allowed label degree. Only used when type == 'set'. Should be between 0 and 1

Variables
n_classes (*int*) – The number of unique classes in y

fit_transform(*X*, *y*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

X
[array-like of shape (n_samples, n_features)] Input samples.

y
[array-like of shape (n_samples,) or (n_samples, n_outputs), default=None] Target values (None for unsupervised transformations).

****fit_params**
[dict] Additional fit parameters.

X_new
[ndarray array of shape (n_samples, n_features_new)] Transformed array.

INDEX

R

T

W