Uncertainty Estimate with SVM

Release 0.0.1

Marko Djordjic

Nov 30, 2019

Contents:

CHAPTER 1

ucf Module

Uncertain estimate with SVM is a package which provides a functional example on how to train a SVC which outputs an uncertainty estimate alongside prediction of class membership.

As noted by Murphy (2012, p. 497) are not probabilistic models. The

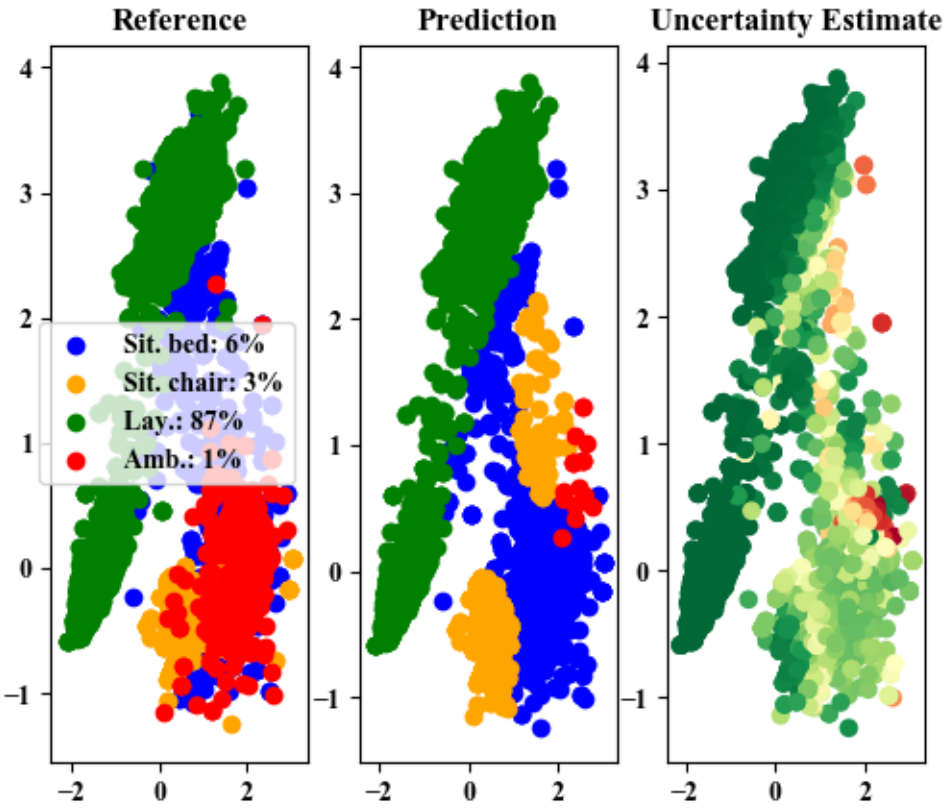
In this specific case, a classification task is solved with a SVC. The same methodology can be extended to solving regression problems as well. In the case of solving a classification task, a specific approach towards building a training set has been applied. Each of the estimators is trained on a subset of training data set. This subset is in itself designed to contain all classes in equal proportions. Therefore the size of the subset is determined by the size of least frequent class and the number of classes.

In order to obtain uncertainty estimate from an otherwise non-probabilistic model, a variational inference approach was utilized.

Examples

```
>>> import uncertainty_estimate_with_svm.ucf as ue_svm
>>> # Get the training set with the equal number of classes.
>>> reduced_set = ue_svm.reduce_set_to_equal_distribution_of_classes(
        features_for_training=features, targets_for_training=targets
        )
>>> ensemble = ue_svm.generate_ensemble(number_of_estimators=30,
        features_for_training=reduced_set[, 0:9],
        targets_for_training=reduced_set[, 10]
        )
>>> predictions, uncertainty = ue_svm.generate_predictions(
        inventory_of_estimators=ensemble,
        features=x_test
    )
```

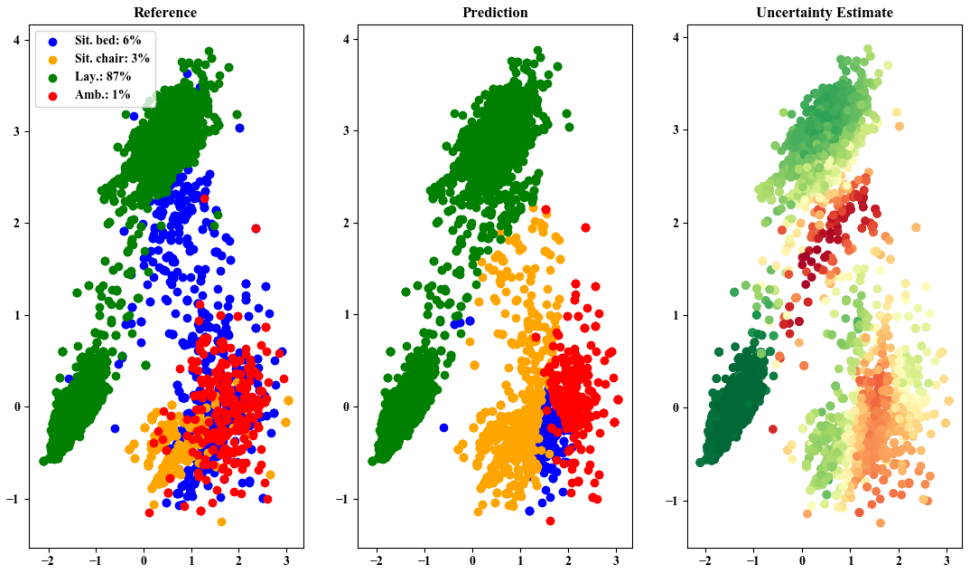
Below we can take a look at the solution produced by the single SVM.



And here is the confusion matrix for the single SVM.

Reference	Sitting on bed	Sitting on chair	Lying	Ambulating
Sitting on bed (n	66	12	1	22
Sitting on chair (	26	69	0	5
Lying (n=13175)	1	0	99	0
Ambulating (n=1	57	0	0	42

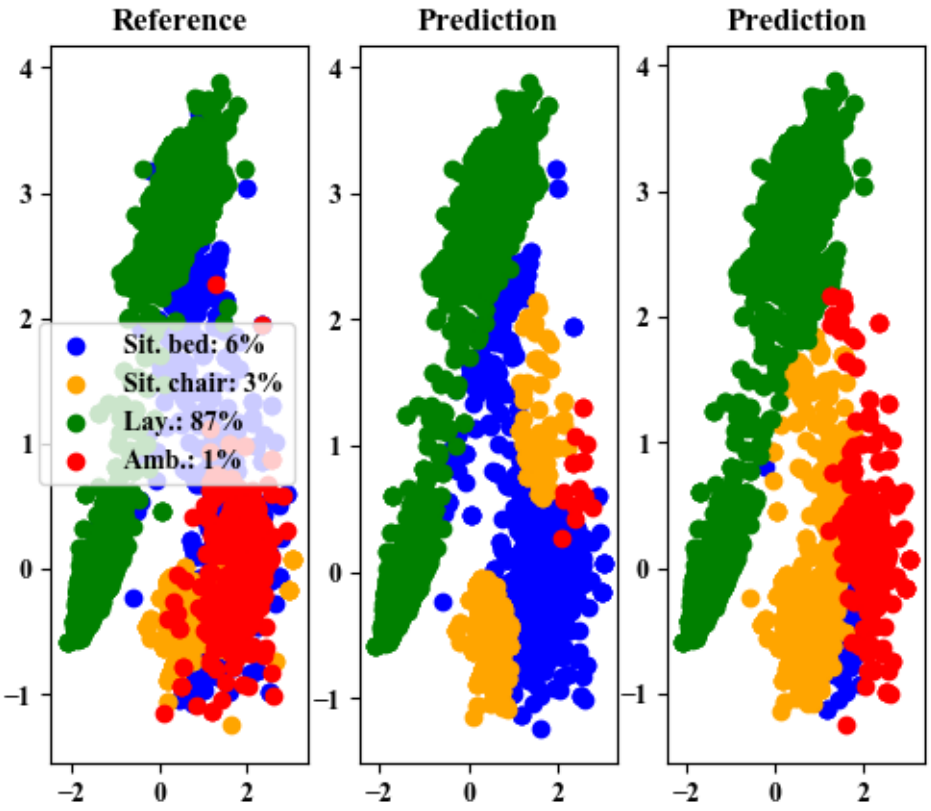
SVC ensemble produces different solution, as we can see from the image below.



Also distribution of classes across the prediction is different.

Reference	Sitting on bed	Sitting on chair	Lying	Ambulating
Sitting on bed (n=1028)	73	6	0	21
Sitting on chair (n=526)	46	46	1	8
Lying (n=13175)	1	0	99	0
Ambulating (n=297)	46	9	0	45

Finally we can take a look a te comparison between a single SVC and an ensemble of SVCs.



There is clearly a different fit, which has been achieved on the basis of training of multiple SVCs on a balanced set, and some of the bias of the model trained on the imbalanced set has be removed. However, even though this

allows for training of SVMs on larger training sets, there are two relative problems with this solution: (a) minor but still present loss in accuracy, and (b) slower execution of the ensemble.

References

Notes

1.1 Functions

compute_predictive_entropy(probability)	Estimate epistemic uncertainty via predictive entropy [1]_
generate_ensemble(number_of_estimators, ...)	Generates a collection of estimators
generate_predictions(...)	Generate predictions from ensemble
get_all_files_within_folder(path, ...)	Catalogue all files within a folder according to negative condition
get_all_sub_folders_within_folder(path)	Catalogue names of all sub-folders within folder
get_data(folder_wh_data)	Build an inventory of data sets out of individual files within a folder.
make_confusion_matrix(reference, output, ...)	Compute confusion matrix
plot_comparison(coordinates, reference, ...)	Plot comparison across different classification solutions.
plot_confusion_matrix(content[, save_plot, path])	Plot confusion matrix
plot_individual_classes(coordinates, ...[, ...])	Plot class membership in individual plot
plot_solution(coordinates, original_labels, ...)	Plotting of solution of classification task
reduce_set_to_equal_distribution_of_classes(...)	Reduces the training set to the size N = K x size of the least frequent class
scatter_plot_with_groups(coordinates, ...[, ...])	Produce scatter plot with coloration according to the labels

1.1.1 compute_predictive_entropy

ucf.**compute_predictive_entropy**(*probability*)

Estimate epistemic uncertainty via predictive entropy[1]

Parameters **probability** (*numpy.array*) – A numpy.array (N x C) with the probabilities obtained from the underlying classier (soft voting).

Returns Uncertainty estimate for each prediction.

Return type numpy.array

Notes

For the computation of uncertainty value equal to zero are replaced with a small constant near zero.

References

1.1.2 generate_ensemble

ucf.**generate_ensemble**(*number_of_estimators*, *features_for_training*, *targets_for_training*)

Generates a collection of estimators

[1] Further details about about predictive entropy available at: *https://en.wikipedia.org/wiki/Entropy_(information_theory)*

Each estimator is trained on a sub-set of the training data, and appended to the ensemble. Support Vector Classifier has been selected as the classifier of choice, but can be replaced with any other classifier.

Parameters

- **number_of_estimators** (*int*) – How much estimators will be in the ensemble.
- **features_for_training** (*numpy.array*) – Features which will be utilized for training of individual estimators.
- **targets_for_training** (*numpy.array*) – Targets which will be utilized for training of individual estimators.

Returns A collection of SVCs trained on different sections of features and targets pairs.

Return type List

Notes

The function does not shuffle the data. If shuffling is necessary, it has to be done before call to the function.

1.1.3 generate_predictions

ucf.**generate_predictions**(*inventory_of_estimators*, *features*)
Generate predictions from ensemble

The function applies 'predict_proba' method to a collection of estimators, in order to get predictions and compute uncertainty estimate via predictive entropy.

Parameters

- **inventory_of_estimators** (*list*) – A collection of estimators placed in a list.
- **features** (*numpy.array*) – Features on which to perform prediction.

Returns

- **ensemble_predictions** (*numpy.array*) – Prediction of class membership.
- **uncertainty_estimate** (*numpy.array*) – Uncertainty estimate.

1.1.4 get_all_files_within_folder

ucf.**get_all_files_within_folder**(*path*, *negative_condition*)
Catalogue all files within a folder according to negative condition

Parameters

- **path** (*str*) – Path to the folder.
- **negative_condition** (*str*) – Exact text contained with the name of the files, which is utilized to identify files which will not be catalogued.

Returns Names of all files within folder except file names designated in *negative_condition* parameter.

Return type list

1.1.5 get_all_sub_folders_within_folder

ucf.**get_all_sub_folders_within_folder**(*path*)
Catalogue names of all sub-folders within folder

Parameters **path** (*str*) – Path to the main folder.

Returns Names of all sub-folder within folder designated in *path* parameter.

Return type list

1.1.6 get_data

ucf.**get_data**(*folder_wh_data*)

Build an inventory of data sets out of individual files within a folder.

Function omits files which are having *txt* inside their name.

Parameters **folder_wh_data** (*str*) – Folder in which data files are residing.

Returns All the data sets generated from individual data files.

Return type list

1.1.7 make_confusion_matrix

ucf.**make_confusion_matrix**(*reference*, *output*, *prediction_labels*)

Compute confusion matrix

Parameters

- **reference** (*numpy.array*) – A vector with reference.
- **output** (*numpy.array*) – A vector with targets.
- **prediction_labels** (*list*) – Descriptions of labels.

Returns

Return type Confusion matrix as the pandas.DataFrame.

Notes

Reference group is placed in row. Proportion of each prediction within the reference group is computed across columns (horizontally).

1.1.8 plot_comparison

ucf.**plot_comparison**(*coordinates*, *reference*, *solutions*, *description*, *coloration*, *save=True*, *path=None*)

Plot comparison across different classification solutions.

Parameters

- **coordinates** (*numpy.array*) – Coordinates of the points.
- **reference** (*numpy.array*) – Labels of the reference.
- **solutions** (*list*) – List containing solutions of classification problem.
- **description** (*dict*) – Description of each label.
- **coloration** (*dict*) – Vector with coloration.

Returns No explicit return. Plot is displayed on the screen, or saved into a file.

Return type None

1.1.9 plot_confusion_matrix

ucf.**plot_confusion_matrix**(*content*, *save_plot=False*, *path=None*)

Plot confusion matrix

Confusion matrix will always be tabulated and plotted. Optionally, picture of confusion matrix can be saved.

Parameters

- **content** (*numpy.array*) – Numpy array with the complete content of the confusion matrix.
- **save_plot** (*bool*) – Indication whether to save the plot. Default set to false.
- **path** (*str*) – Path including the file name where to save the plot.

Returns No explicit return. Optionally plot can be saved.

Return type None

Notes

Content must consist all numeric content, as well as headings of all rows and columns.

1.1.10 plot_individual_classes

ucf.**plot_individual_classes**(*coordinates*, *class_membership*, *description*, *coloration_mode*, *coloration*, *save=False*, *path=None*)

Plot class membership in individual plot

Point within scatter plots indicating class membership with multiple classes can often overlap, therefore debilitating correct analysis. This function plots all classes independently.

Parameters

- **coordinates** (*numpy.array*) – Coordinates of points.
- **class_membership** (*numpy.array*) – Indication of class membership.
- **description** (*dict*) – Description of each label.
- **coloration_mode** (*str*) – Indication of the mode of coloration.
- **coloration** (*numpy.array*) – Vector with coloration.
- **save** (*bool*) – Option to save the plot.
- **path** (*str*) – Absolute path towards the file in which to save the plot.

Returns No explicit return.

Return type None

1.1.11 plot_solution

ucf.**plot_solution**(*coordinates*, *original_labels*, *predicted_labels*, *legend_colors*, *legend_descriptions*, *uncertainty*, *save=False*, *path=None*)

Plotting of solution of classification task

Convenience function to plot: (a) original labels, (b) predicted labels, and (c) uncertainty estimate of the model.

Parameters

- **coordinates** (*numpy.array*) – Coordinates of labels

- **original_labels** (*numpy.array*) – Reference one-dimensional encoding of the class membership. One-hot encoding is not supported.
- **predicted_labels** (*numpy.array*) – Predicted one-dimensional encoding of the class membership. One-hot encoding is not supported.
- **legend_colors** (*dict*) – Colors to be utilized for coloration of points.
- **legend_descriptions** (*dict*) – Labels to be utilized for description in plot legend.
- **uncertainty** (*numpy.array*) – Uncertainty of the models estimate of class membership.
- **save** (*bool*) – Option to save the plot. Default set to false.
- **path** (*str*) – Absolute path to the file in which to save a plot.

Returns No explicit return. Plot is displayed on the screen, or saved into a file.

Return type None

1.1.12 reduce_set_to_equal_distribution_of_classes

ucf.**reduce_set_to_equal_distribution_of_classes**(*features_for_training*, *targets_for_training*)

Reduces the training set to the size N = K x size of the least frequent class

Firstly, the count of least frequent class is computed. Than a pair with features and targets is constructed consisting of samples of all classes. Therefore, generated pair is balanced in regards to distribution of classes.

Parameters

- **features_for_training** (*numpy.array*) – Features which will be used for generating reduced sets.
- **targets_for_training** (*numpy.array*) – Targets which will be used for generating reduced sets.

Returns Two separate numpy.arrays Features and targets reduced to the size of equal to the number of samples belonging to the leas frequent class.

Return type numpy.array

1.1.13 scatter_plot_with_groups

ucf.**scatter_plot_with_groups**(*coordinates*, *labels*, *legend_colors*, *legend_descriptions*, *save_plot=False*, *path=None*)

Produce scatter plot with coloration according to the labels

Parameters

- **coordinates** (*numpy.array*) – Coordinates of points.
- **labels** (*numpy.array*) – Vector indicating class membership of each point.
- **legend_colors** (*dict*) – Colors to be utilized for coloration of points.
- **legend_descriptions** (*dict*) – Labels to be utilized for description in plot legend.
- **save_plot** (*bool*) – Indication whether to save a plot. Defaults to none
- **path** (*str*) – Path including the file name where to save the plot.

Returns No explicit return.

Return type None

1.2 Classes

PCA([n_components, copy, whiten, ...])	Principal component analysis (PCA)
SVC([C, kernel, degree, gamma, coef0, ...])	C-Support Vector Classification.
TrainingDataSets(features_and_targets_data_set)	Class for generating training, validation, and testing data sets.
product	product(*iterables, repeat=1) –> product object

1.2.1 TrainingDataSets

class ucf.**TrainingDataSets**(*features_and_targets_data_set*)

Bases: object

Class for generating training, validation, and testing data sets.

original_data
Original features and targets.

Type pandas.DataFrame

indices_of_features
Numeric indication of position of features in *original_data*

Type list

indices_of_targets
Numeric indication of position of targets in *original_data*

Type list

train_features
Unscaled training features.

Type numpy.array

train_targets
Training targets. Shuffled if desired.

Type numpy.array

validation_features
Validation features

Type numpy.array

validation_targets
Validation Targets

Type numpy.array

test_features
Testing features

Type numpy.array

test_targets
Testing targets

Type numpy.array

scaled_train_features
Train features scaled to mean zero and unit variance. Shuffled if desired.

Type numpy.array

scaled_validation_features
Validation features scaled to mean zero and unit variance.

Type Numpy Array

scaled_test_features
Test features scaled to mean zero and unit variance.

Type Numpy Array

Methods Summary

compute_mean_and_standard_deviation()	Computation of mean and standard deviation of features in the training data set.
get_scaled_features()	Convenience method to return scaled features.
get_targets()	Convenience method to return targets and features
make_training_data(train_size, validation_size)	Make features and targets
scale_features()	Standardize features in such manner that their mean is centered to zero, and unit of measurement is set to variance.
shuffle()	Shuffle scaled features and unscaled targets for training

Methods Documentation

compute_mean_and_standard_deviation()
Computation of mean and standard deviation of features in the training data set.

Returns No explicit return.

Return type None

Notes

Values are stored in the 'features_mean' and 'features_standard_deviation' attribute of the class.

get_scaled_features()
Convenience method to return scaled features.

Returns

- **scaled_train_features** (*numpy.array*) – Scaled features for training.
- **scaled_validation_features** (*numpy.array*) – Scaled features for validation.
- **scaled_test_features** (*numpy.array*) – Scaled features for testing.

get_targets()
Convenience method to return targets and features

Returns

- **train_targets** (*numpy.array*) – Targets for training.
- **validation_targets** (*numpy.array*) – Targets for validation.
- **test_targets** (*numpy.array*) – Targets for testing.

make_training_data(*train_size*, *validation_size*)
Make features and targets

Parameters

- **train_size** (*int*) – Proportion of the training set.
- **validation_size** (*int*) – Proportion of the validation set.

Notes

Size of the testing set is determined implicitly.

scale_features()
Standardize features in such manner that their mean is centered to zero, and unit of measurement is set to variance.

Returns Standardized features are placed inside appropriate attributes of the class.

Return type numpy.array

shuffle()
Shuffle scaled features and unscaled targets for training

Notes

Only scaled training features and targets are shuffled. Validation, and test data sets are not shuffled.

1.3 Class Inheritance Diagram

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

u