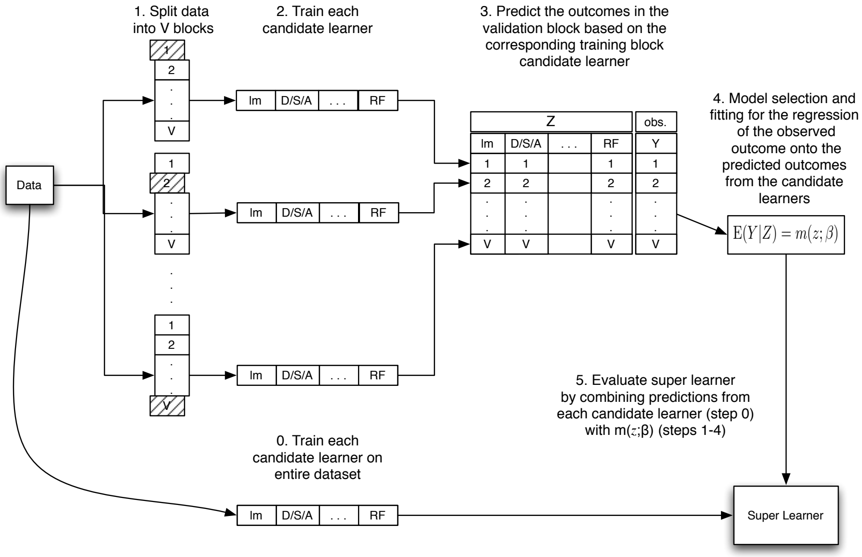

1. Split data into V blocks
2. Train each candidate learner
3. Predict the outcomes in the validation block based on the corresponding training block candidate learner
4. Model selection and fitting for the regression of the observed outcome onto the predicted outcomes from the candidate learners
5. Evaluate super learner by combining predictions from each candidate learner (step 0) with m(z;β) (steps 1-4)
0. Train each candidate learner on entire dataset
Data
1
2
V
lm
D/S/A
. . .
RF
Z
obs.
Y
$\mathrm{E}(Y|Z) = m(z;\beta)$
Super Learner

COMP47590

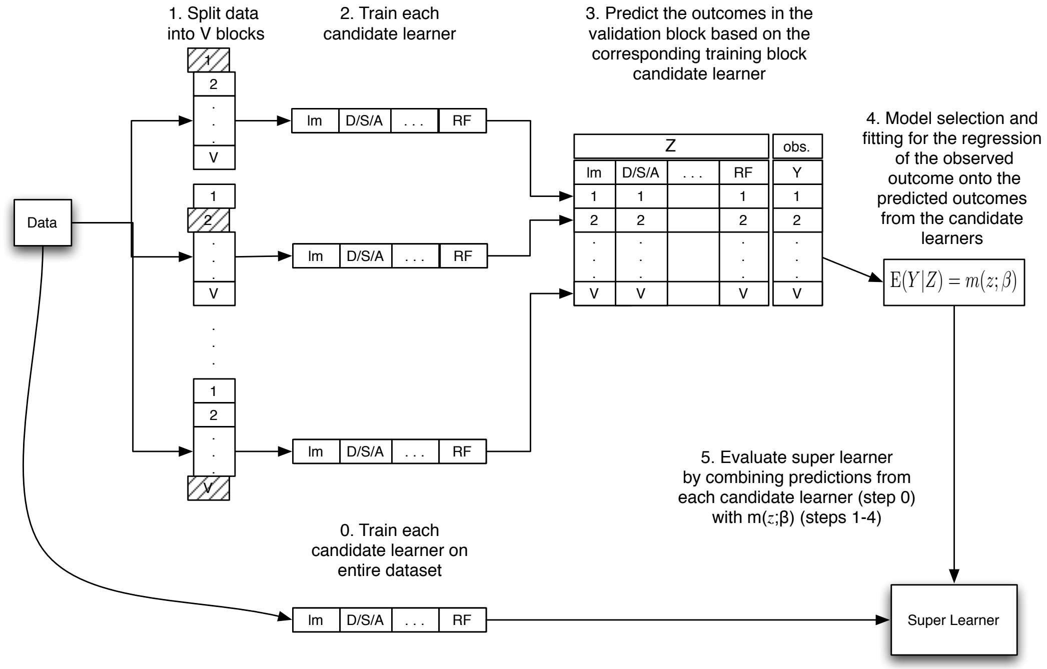
Advanced Machine Learning

Assignment 1: The Super Learner

Introduction

The *Super Learner* is a heterogeneous stacked ensemble classifier. This is a classification model that uses a set of base classifiers of different types, the outputs of which are then combined in another classifier at the stacked layer. The *Super Learner* was described in (van der Laan et al, 2007) but the stacked ensemble idea has been around for longer than that - for example (Wolpert, 1992).

Figure 1 shows a flow diagram of the Super Learner process (this is from (van der Laan et al, 2007) and the process is also described in the COMP47590 lecture "COMP47590 2017-2018 L04 Supervised Learning Ensembles 3"). The base classifiers are trained and their outputs are combined along with the training dataset labels into a training set for the stack layer classifier. To avoid overfitting the generation of the stacked layer training set uses a *k*-fold cross validation process (described as *V*-fold in Figure 1). To further add variety to the base estimators a bootstrapping selection (as is used in the bagging ensemble approach).



1. Split data into V blocks
2. Train each candidate learner
3. Predict the outcomes in the validation block based on the corresponding training block candidate learner
4. Model selection and fitting for the regression of the observed outcome onto the predicted outcomes from the candidate learners
5. Evaluate super learner by combining predictions from each candidate learner (step 0) with m(z;β) (steps 1-4)
0. Train each candidate learner on entire dataset
Data
1
2
V
lm
D/S/A
...
RF
Z
obs.
Y
$E(Y|Z) = m(z;\beta)$
Super Learner

Figure 1: A flow diagram for the Super Learner

The task in this assignment is to implement the Super Learner classifier as an extension to scikit-learn. Although you can use scikit-learn base estimator implementations (e.g. decision trees, logistic regression, or support vector machine models) you must implement the Super Learner yourself.

Tasks

Perform the following tasks:

1. Write a SuperLearnerClassifier Python class that implements the Super Learner algorithm with the following characteristics:
    - A fixed set of 6 base models should be established using existing implementations from scikit-learn (e.g. decision trees, logistic regression, or support vector machine models). Remember that the stacked ensemble approach works best with heterogeneous models.
    - It is reasonable to set sensible default hyper-parameters for these base estimators.
    - Generate the stacked layer training set using the label outputs from the base estimators.
    - At the stacked layer use a decision tree model.
2. Evaluate the performance of the Super Learner using a 10-fold cross validation on the MNIST fashion training dataset available from Kaggle at https://www.kaggle.com/zalando-research/fashionmnist .
    - Use the scikit-learn function **`sklearn.model_selection.cross_val_score`** to perform the cross validation.
    - Think about the most appropriate performance measure for the evaluation.
3. Modify the implementation so that the SuperLearnerClassifier constructor can take a parameter indicating whether the stacked layer classifier should be trained on label outputs from the base classifiers or probability outputs from the base classifiers.
    - Most base estimators can generate a probability distribution across possible classes using the **`predict_proba`** function.
4. Modify the implementation so that the SuperLearnerClassifier constructor can take a parameter to specify the type of model to use at the stack layer
    - It is sensible to limit the types of estimators that the algorithm can use at the stack layer to a sensible set of 5 - 10 algorithms from scikit-learn.
    - Make sure to include the **`sklearn.tree.DecisionTree`** and **`sklearn.linear_models.LogisticRegression`** algorithms as choices.
5. Perform an evaluation experiment comparing the performance of the model using label outputs to train the stack layer and a model using probability outputs to train the stack layer.
    - Choose an appropriate experimental method and performance measure.
    - Experiment with both decision trees and logistic regression models at the stack layer.
    - Keep the base estimators specification and any other parameters constant for this experiment.

6. Modify the implementation so that the SuperLearnerClassifier constructor can take a parameter to specify the base estimators to use.
    - This could be reasonably specified in different ways - for example, as a specific list of exactly the base estimator model types to use or a list of allowed base estimator types and an overall number of base estimators in the ensemble to be created based on this list.
    - It is sensible to limit the types of base estimators that the algorithm can use to a sensible set of 5 - 10 algorithms from scikit-learn.
    - It is not required to also specify hyper-parameters for each algorithm, use a default set.
7. Perform a grid search to determine the best setup (e.g. base estimators, stack layer model, stack layer training data type, etc) for the Super Learner.
8. It has been suggested that also adding the original input to the input at the stack layer could improve the Super Learner. Implement a modification that includes this and evaluate its impact.
    - Use the best setup found in Task 7.
9. Ensemble models rely on the base estimators being strong predictors but at the same time weakly correlated with each other. Implement some functionality to show the extent to which the estimators in your model satisfy these criteria?

Notes

The following notes may be useful:

- **Can I Use Scikit-Learn And Other Python Packages?** One of the goals of this assignment is to gain experience writing original machine learning algorithms, rather than simply using existing packages. We recognise, however, that very good implementations of machine learning algorithms exist in packages like scikit-learn and re-implementing things needlessly is often of limited value. So, for this assignment we seek to strike a balance. You must write your own Super Learner implementation but can use scikit-learn implementations for the base estimators and stack layer estimators. You should also use scikit-learn functions for performing tasks like cross validation and grid searches. Similarly all scikit-learn models are built on top of **numpy** and it provides many useful utility functions. You should, however, write the core super Learner algorithm yourself. If in doubt about the use of a package please just ask.
- **It's Taking Forever!** While the MNIST Fashion dataset isn't enormous it is big enough to start to cause problems once we start performing cross validations and grid searches. If you find things are taking too long feel free to down-sample the dataset. A 10% sample still leaves 6,000 rows. Submissions will not be penalised for using down-sampled datasets. Ensure, however, that the target feature levels remain stratified.
- **Do I Need to Submit Lots of Versions of SuperLearnerClassifier?** No! One implementation that includes all functionality is fine. There is no need to show the different versions of the implementation as you move through the different tasks. The full notebook should run, however.

Submission

The key submission details for the assignment are as follows:

- **Submission date:** Thursday 8th March 2018 before 23:59
- **Submission method:** Submissions should be made through the module Moodle site
- **Submission format:** Submissions should compose a zip file containing the following:
    - a completed version of the template Jupyter notebook provided in .ipynb format (the Python notebook and code should include adequate comments to understand your implementation)
    - a html export of your Jupyter notebook after execution that contains all output
    - any other files required to execute your code.
- **Late submissions:** Late submissions will be penalised at 5% penalty per day.

Marking

Marking of tasks will be based on the following weighting.

Task 1 & 2	30%	Implement & evaluate the SuperLearnerClassifier
Task 3, 4 & 5	20%	Implement & compare alternative stack layer approaches
Task 6	15%	Implement base estimator specification interface
Task 7	10%	Perform and interpret grid search
Task 8	10%	Evaluate the impact of adding original descriptive features at the stack layer
Task 9	15%	Add and demonstrate functionality for interrogating performance and correlation of base models

References

van der Laan, M., Polley, E. & Hubbard, A. (2007). Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1) doi:10.2202/1544-6115.1309
https://pdfs.semanticscholar.org/19e9/c732082706f39d2ba12845851309714db135.pdf

Wolpert, D.H., Stacked generalisation, Neural Networks 5 (1992) 241-259.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.1533