Estimation, Detection and Learning II

Improvement Techniques

Catarina Oliveira

DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UNIVERSIDADE PORTUCALENSE

CONTENT

DEPARTAMENTO CIÊNCIA E TECNOLOGIA

Ensemble Learning

Ensemble Learning

Predictive performance in classification tasks can be improved by combining predictions from multiple models.

→ *Ensemble of classifiers*

- **Homogeneous** : models created with the same technique
- **Heterogeneous** : models created with different techniques

Base classifier **:**

- *classifier* whose predictions are combined in the *ensemble*
- Each can be created using
  - the original *trainset*
  - Parts of the original *trainset*

base requirements ensemble *classifiers :*

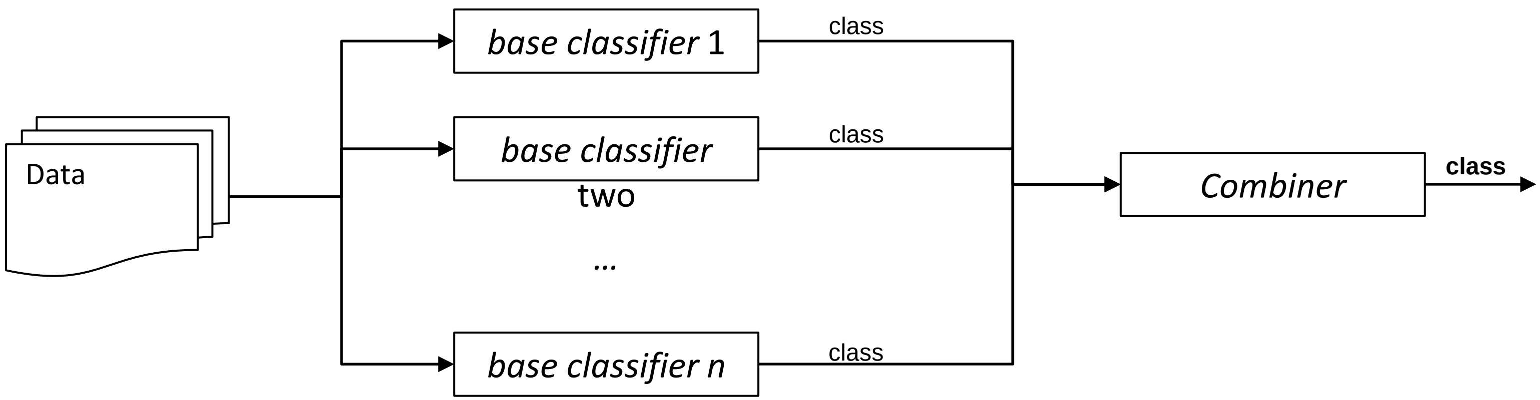
- **Predictive performance** : they must outperform the model that predicts the majority class
- **Predictive diversity** : should be independent, ideally making mistakes on different parts of the data

Approaches:

- Parallel (eg : *bagging* , *random forests*)
- Sequential (eg *AdaBoost*)
- hierarchical

Ensemble learning : parallel approach

- the most common
- Attempts to explore the similarities and differences of predictions made by different *base classifiers*
- each *base classifier*
  - It is induced using instances of the original *trainset*
    - All instances | all *features*
    - Sample instances | all *features*
    - All instances | *features*



Ensemble learning : parallel approach – combination of predictions

Voting : the class predicted by most *classifiers* is the class predicted by the *ensemble*

weighted voting : the class predicted by each base *classifier* is associated with a weight, which represents how much the prediction of this *classifier* should be considered for the final prediction of the *ensemble*

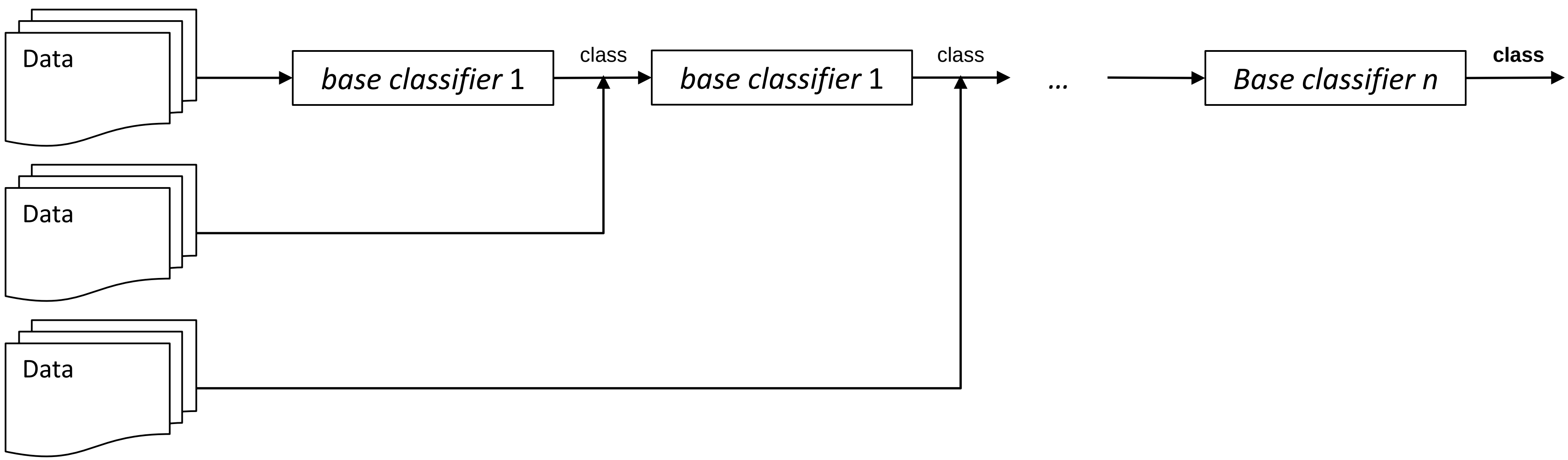
Stacking : a classification algorithm is used to predict the final class of the *ensemble* , having as *features* the predictions of the various base *classifiers*

Ensemble learning : sequential approach

The induction of a base *classifier* uses information from previously induced base *classifiers (eg* , combine predictions of previously induced *base classifiers with features*)

Can be used for:

- Hierarchical sorting tasks
- *multilabel* tasks *classification*



bagging

bagging

- each *base classifier is induced using a trainset*
  - *bootstrap* * approach , have the same number of objects as the *trainset*

- Combines predictions from base *classifiers* by *voting*

- Can be used for <u>unstable classification techniques </u>(*unstable predictors*): Your predictive performance is affected by changes in *trainset composition* . Ex: decision trees, neural networks)

- Robust to *overfitting when there is trainset*

- Number of generated models is a *hyper-parameter* for the *bagging technique*
  - The higher, the lower the prediction variance (and the higher the computational cost)

- It can also be used for regression
  - Combination is done by averaging

- **Results:**
  - Forecasts
  - *base models* generated

* Next chapter

Bagging : definition of *hyper-parameters*

- **Number of *base models*** to generate
  - The bigger the better
    - Paying attention to the computational cost
  - Generally, 100 is considered a good choice.

- ***base learner*** to use to generate the models
  - Some approaches use decision trees
  - Others allow the user to choose the *base learner*
  - Most common: decision trees, neural networks (due to their instability)

Baggage : advantages and disadvantages

Benefits	Disadvantages

• Improves the predictive performance of the *base learner*
  • since this is an *unstable predictor*
• Few *hyper-parameters* to define
	• *bootstrapping* sampling has a random component
	  • but the variability of the results can be minimized by choosing the number of *base models* to generate
	• Computationally more "expensive" than using a simple model
	  • But it can be run in parallel

Random Forests

Random Forests

- Combine multiple decision trees
- Similar to *bagging* : Each decision tree is created with a different bootstrapped *sample*
- Different from *bagging* : at each node of the tree, instead of choosing the *split* from all *features* , only a predefined number of randomly selected attributes are used
- Good choice for *datasets* with many *features*
- **Results:**
  - Forecasts
  - Statistics on the importance of *features*

DEPARTAMENTO CIÊNCIA E TECNOLOGIA

Random Forests : definition of *hyper-parameters*

- **Number of *base models*** to generate
  - Recommended: 1000
  - To get more reliable statistics on the importance of *features* : 5000

- **number of *features*** to choose randomly at each node
  - depends on the problem
  - Rule of thumb: $\sqrt{\#features}$

DEPARTAMENTO CIÊNCIA E TECNOLOGIA

Random Forests : advantages and disadvantages

Benefits	Disadvantages

• Good predictive performance in several problems
• Relatively easy to interpret
• Easy to define *hyper-parameters*
	• Computationally “expensive”
	  • Because the recommended number of trees is high
	    • But it can be run in parallel
	• randomness
	  • Can be minimized by using the minimum recommended number of trees

DEPARTAMENTO CIÊNCIA E TECNOLOGIA

Boosting

Boosting : generic algorithm

1. the *base learner* assigns all instances equal weights

2. Repeat up to *base learner limit* be achieved, or the predictive performance increases:

    1. If there is any prediction error caused by the first *base learner* , the weight of the errored observations is increased

    2. Apply next *base learner*

Boosting : AdaBoost

- One of the most representative *boosting*

- In each training iteration, a *base classifier* is induced using the *trainset* and each instance is assigned a weight according to how well the model predicted its class
- The more difficult the class prediction, the greater the weight associated with the instance

- The weight of an instance defines the probability of being chosen for the *trainset of the* next (sequential) base *classifier*

- Good to use with *weak classifiers* : predictive performance only slightly better than random prediction

- Can be used for regression: *gradient Boosting* , *KGBoost*

- **Results:**
  - Forecasts

AdaBoost : definition of *hyper-parameters*

- **number of iterations**
  - Algorithm authors use: 100

  (*Freund, Y. and Shapire , RE (1996) Experiments with a new boosting algorithm, in Proceedings of the 13th International Conference on Machine Learning, ICML 1996, pp. 148–156.*)

AdaBoost : advantages and disadvantages

Benefits	Disadvantages

• Good predictive performance in several problems
• Easy to define *hyper-parameters*
	• Computationally “expensive”
	  • Because the number of models generated depends on the number of iterations
	  • Cannot run in parallel (sequential)
	• hard to interpret

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.