Advanced Data Analysis

UE Master 2 SCI20

N. Usunier nicolas.usunier@utc.fr

Overview of this lecture

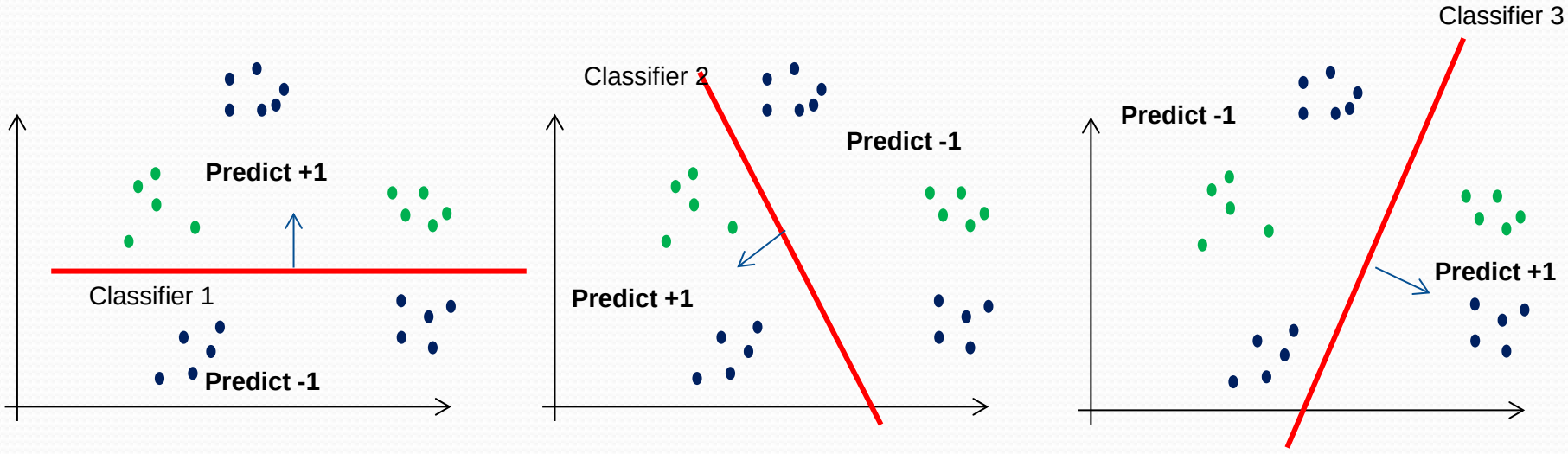
- Lecture IV:
  - Combining prediction functions
  - Bagging and Random Forests

Combining prediction functions

Introduction and motivation

- A classification problem:

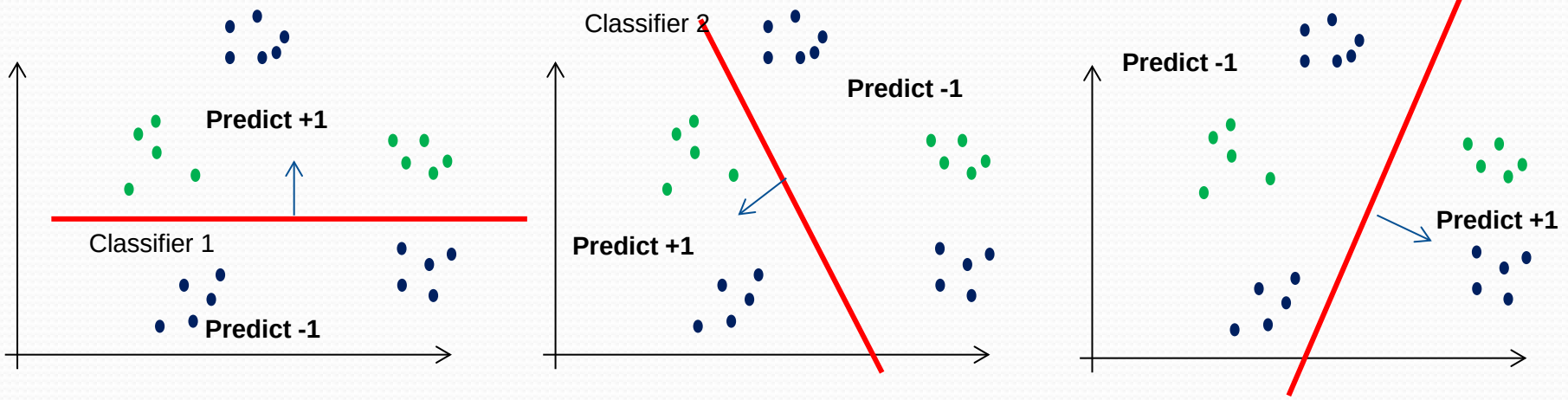
  Can we combine several classifiers to obtain better performance than the best of them ?

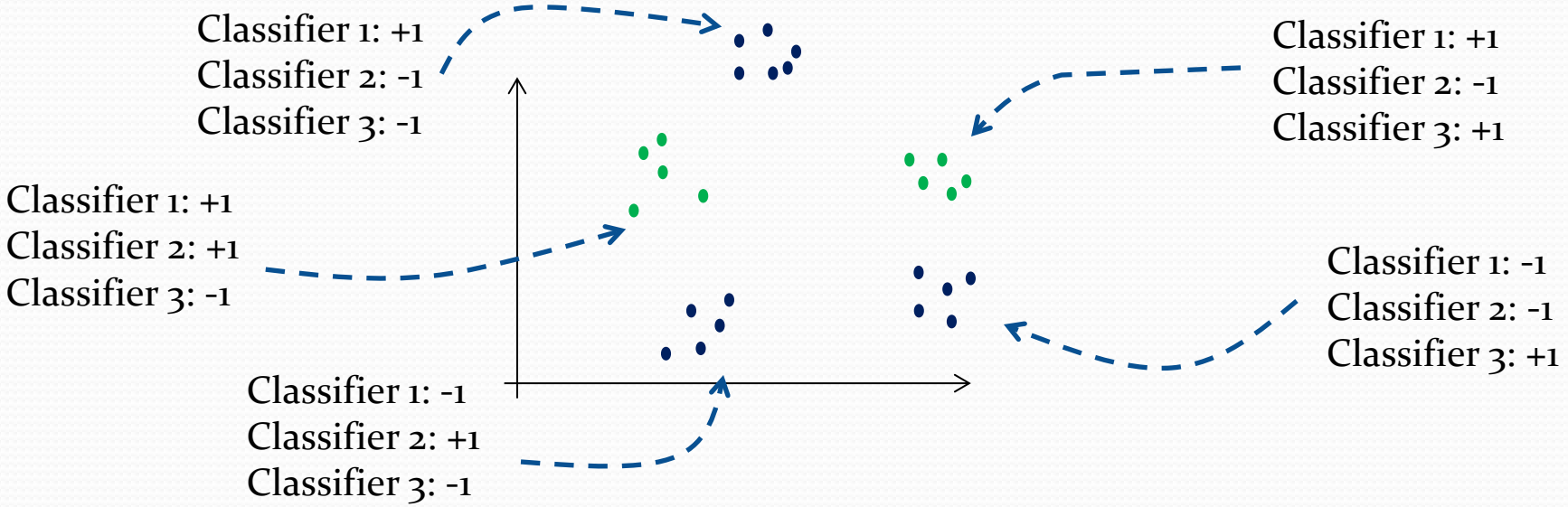


Training data:
Green dots (+1)
Blue dots (-1)

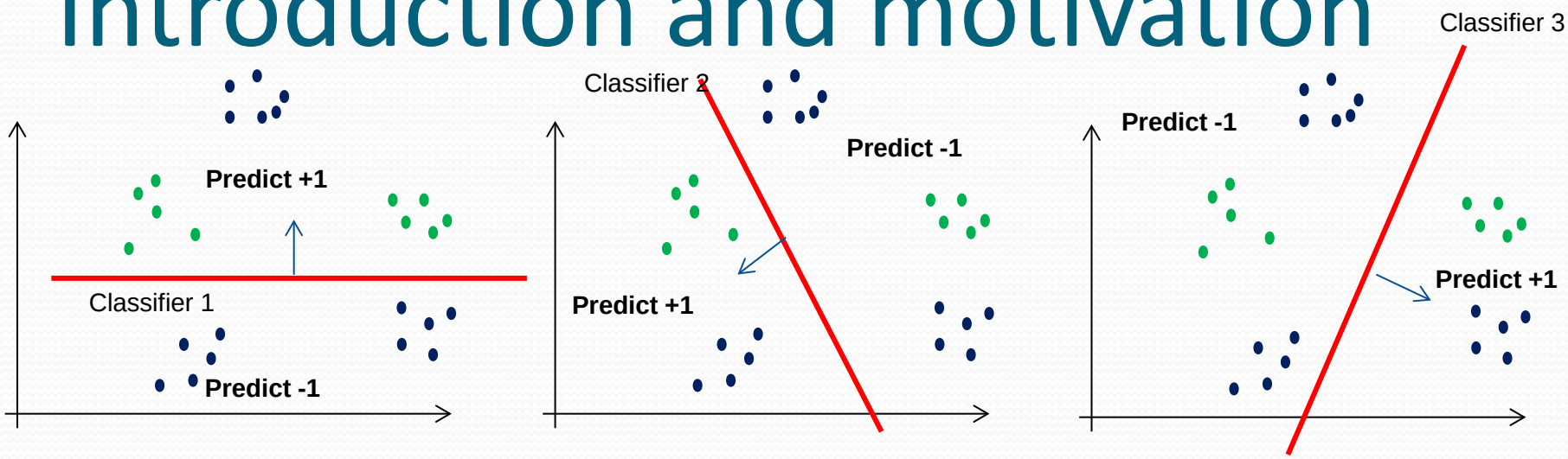
Data not linearly separable
Each linear classifier has error rate 1/5

Introduction and motivation





Introduction and motivation



Classifier 1: +1
Classifier 2: -1
Classifier 3: -1

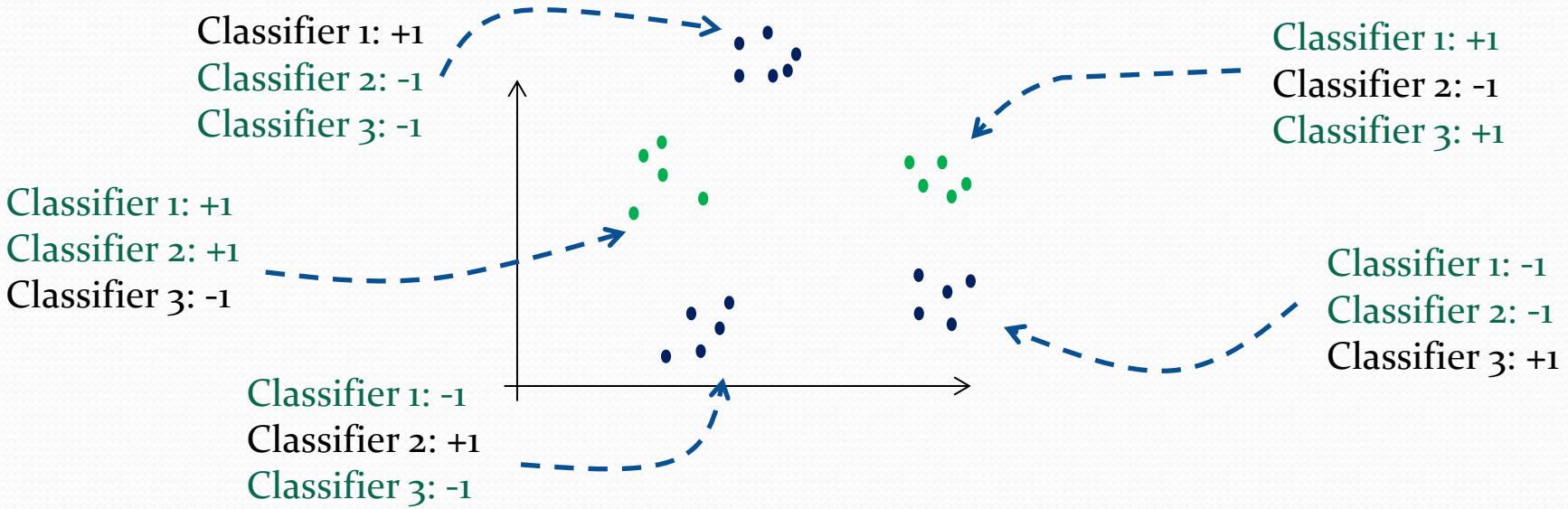
Classifier 1: +1
Classifier 2: -1
Classifier 3: +1

Classifier 1: +1
Classifier 2: +1
Classifier 3: -1

Classifier 1: -1
Classifier 2: -1
Classifier 3: +1

Classifier 1: -1
Classifier 2: +1
Classifier 3: -1

Introduction and motivation

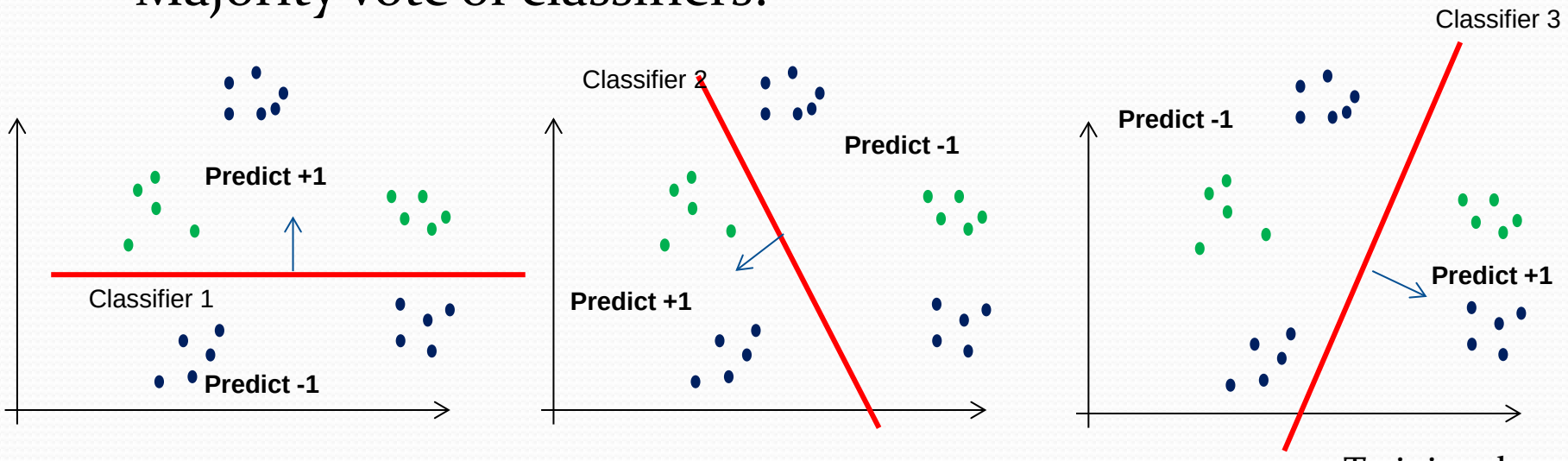


Data is not linearly separable
Each classifier has error rate 1/5

But: taking the majority vote of the 3 classifiers has 0 error rate

Introduction and motivation

- Majority vote of classifiers:



Training data:
Green dots (+1)
Blue dots (-1)

- Each classifier makes 1/5% errors
- The classifiers *do not make the same errors*
- When a classifier is wrong, the other ones are correct

Introduction and motivation

- Combining classifiers:
  - Final prediction: (weighted) vote of the different classifiers
    (e.g. give more weight to the best classifiers)
- Combining regression functions:
  - Final prediction: (weighted) average of the predictions
- If the predictors do not make the same errors:
  combining a lot of predictors improves performances
- If the predictors tend to make the same mistakes:
  combining predictors can decrease performances

Introduction and motivation

- Combining prediction functions 1:
  - Combining prediction functions given by diffferent algorithms
    (e.g. combine (ensembles of) trees, SVMs, nearest neighbors, etc.)
  - Can slightly improve performances because the algorithms do not have the same assumptions on the relathionship between x and y
  - Mainly used in machine learning competitions, less in practice
    - e.g. the 1million $ NetFlix prize for predicting users ratings to movies: combinations of hundreds of models from about ten algorithms

Introduction and motivation

- Algorithms that combine prediction functions:

  Two possible (usually exclusive) goals:

  1. Improve the *training error* of low-performance prediction functions

     Main approach is Boosting (not studied in this lecture)

  1. Improve the *generalization error* of prediction functions

     when we believe the errors are not correlated

     useful for algorithms are *unstable*, i.e. sensitive to small, purely random changes in the training data

     Main approach is Bagging

Bagging

- Bagging = bootstrapping + aggregating
- Bootstrapping:
  - Repeat an experiment on random samples from the training data (sampling with replacement)
  - In bagging: an experiment = learning a classifier/regressor
- Aggregating:
  - Combining the prediction functions learnt on each boostrap sample by voting (classification) or averaging (regression)
- Bagging is a *meta-algorithm*: it takes a learning algorithm as input
  - Any learning algorithm can be used
  - Usually decision or regression trees

Bagging

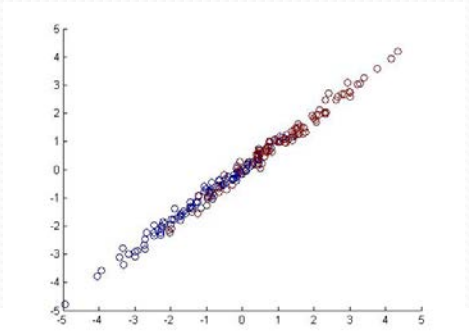
- Bagging:
  - Input: a training set S containing m examples, a number of prediction functions to combine N, a learning algorithm L
  - Algorithm:
  1. Generate N new training sets D_1, …, D_N by sampling with replacement m examples from the training set S
  2. Learn a classifier/regressor on each dataset D_i using algorithm L
  - Output: the combination of the prediction functions

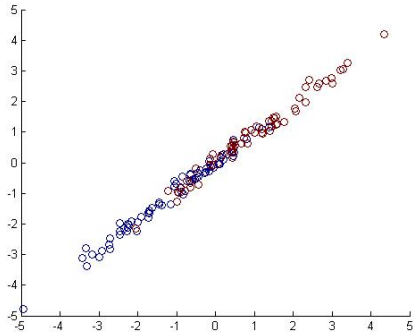
    (majority vote for classification, mean for regression)

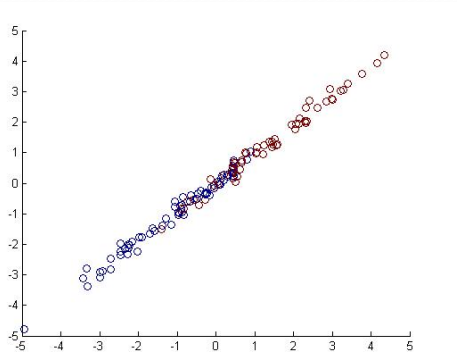
- Notes:
  - By sampling with replacement, some examples of S can appear several times in each D_i
  - As m becomes large, on average each D_i contains $m * 0.632$ unique examples of S

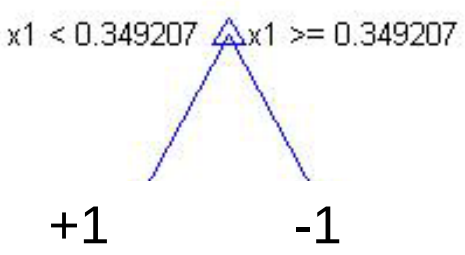
Goal of bagging: making algorithms more stable

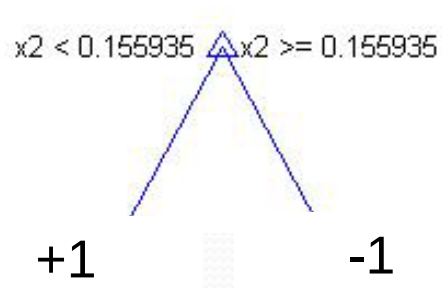
- Unstability of decision/regression trees:
- A simple 2D example with highly correlated features
- Uses Matlab decision trees with pruning level chosen by cross-validation

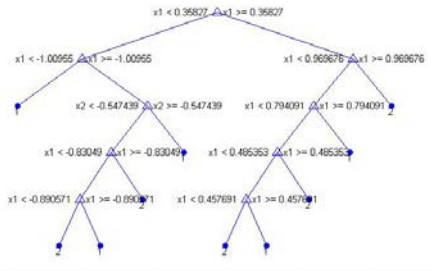












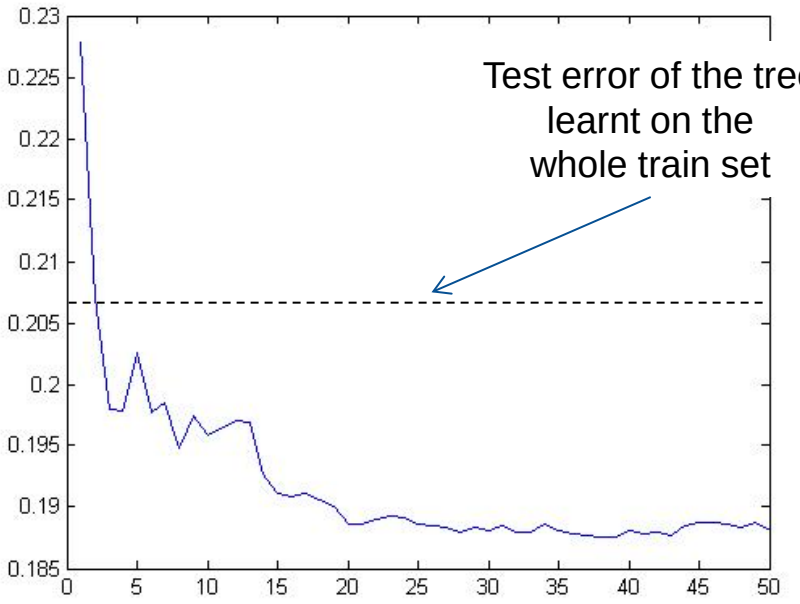
Tree on the whole train set

Tree on a first bootsrap sample

Tree on a second bootsrap sample

When bagging improves performance

- Data from the same distribution as before
  - Using a training set of 1000 examples
  - Test error for bagging from 1 to 50 trees:



As more trees are used, changes in the tree structure due to small, purely random variations of the training data compensate for each other:

Taking the majority vote increases performances, even compared to the tree learnt on the whole train set

Bagging: conclusion

- Bagging allows to correct errors made by unstable algorithms
  - usually works well, typically with trees

    less usefull with more stable algorithms such as linear regression or SVMs
  - but: no theoretical guarantees
- Random Forests: a variation of bagging
  - Uses bagging with decision or regression trees
  - For each bootstrap sample, only considers a random subset of features to introduce controlled variability in the trees
  - Does not use pruning

  (stopping criterion on the number of leaves or the number of examples per leaf)
- Random Forests are used in Microsoft Kinect
  - More generally, a state-of-the-art algorithm for reasonably low dimensional data