Supervised and Experiential Learning

SUPERVISED AND EXPERIENTIAL LEARNING

IDEAI

(Part VI – Ensamble of Classifiers/Multiple Classsifiers /Diversity)

Miquel Sànchez-Marrè

Intelligent Data Science and Artificial Intelligence Research Centre (IDEAI-UPC)

Knowledge Engineering and Machine Learning Group (KEMLG-UPC)

Computer Science Dept.
Universitat Politècnica de Catalunya · Barcelona**Tech**

miquel@cs.upc.edu
http://www.cs.upc.edu/~miquel

Course 2019/2020

https://kemlg.upc.edu

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PART 6 – ENSAMBLE OF CLASSIFIERS,

MULTIPLE CLASSIFIERS, DIVERSITY

https://kemlg.upc.edu

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

CLASSIFIER ENSEMBLE METHODS

https://kemlg.upc.edu

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

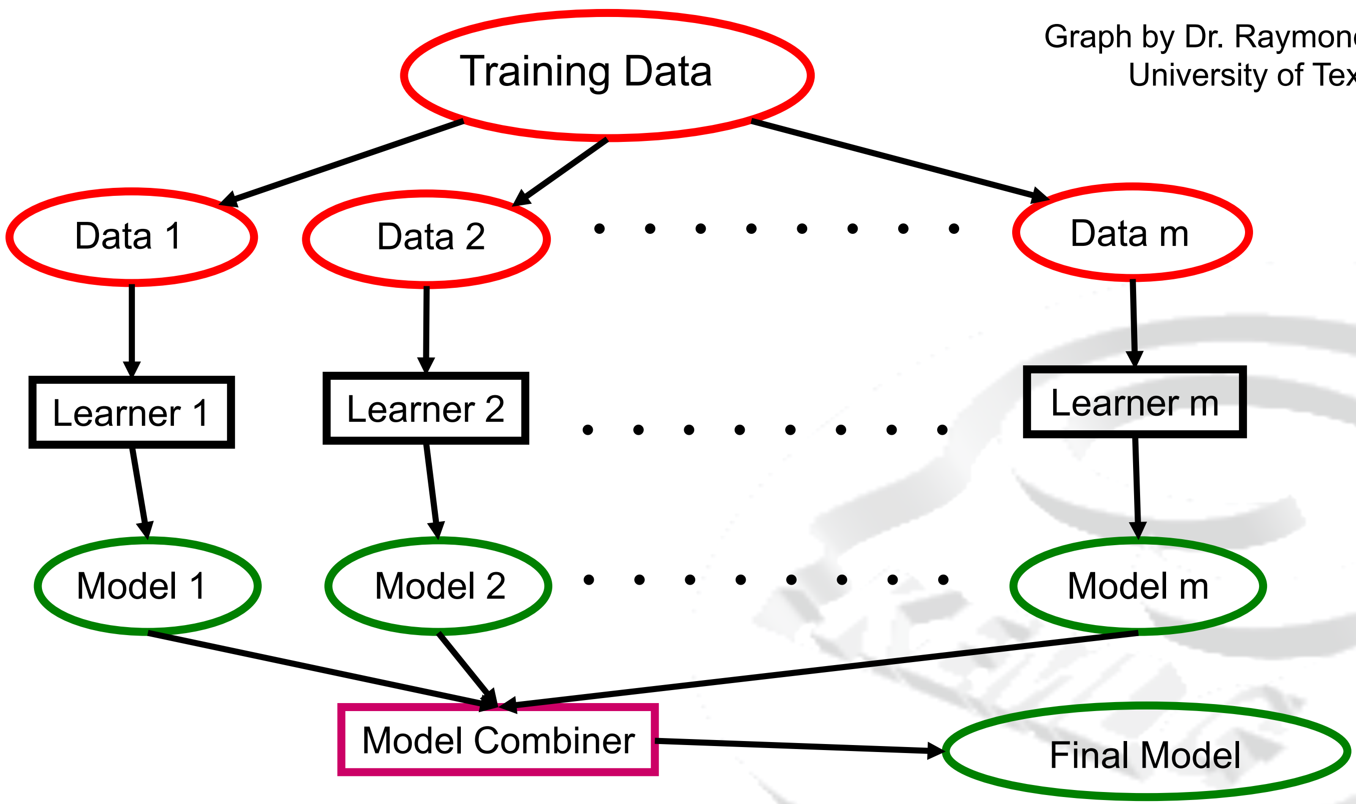
Classifier Ensemble Methods

- **Goal**: to induce a *set of discriminant models* (*classifiers*), which can be of the same type or not, from different samples or weighting the samples or randomly selecting the attributes at each split node of a tree, of a supervised database, aiming at *reducing the discrimination error* of each one of the (weak) classifiers .
- **Applicability criteria**: a *supervised database*, with one qualitative attribute being the *class attribute*, with a representative number of examples of the different possible *class labels*.
- **Most common methods**:
  - Bagging [Breiman *et al*., 1984]
  - Boosting [Shapire, 1990], AdaBoost [Freund & Shapire, 1996]
  - Random Decision Forests [Ho, 1995], Decision Forests [Ho, 1998], Random Forests [Breiman, 2001]
- **Input**: original supervised data matrix
- **Output**: a *set of classifiers* which are able to discriminate/classify the qualitative attribute of interest (class attribute)
- **Evaluation Parameters**: predictive accuracy, scalability, robustness
- **Discrimination/classification process:** when a new instance must be discriminated, *all the classifiers* are used to get their class label prediction. The *class label assigned* to the new instance is *the result of a (weighted) voting among the class labels of the set of classifiers*

Learning Ensembles

- Learn a set of discriminant/classifiers models using *different training data* or/and *different learning algorithms* and/or using other *diversification techniques*
- Combine the output of the classifiers, *i.e.* predicted label, using (weighted) majority voting scheme

Graph by Dr. Raymond J. Mooney
University of Texas at Austin



Value of Ensembles

- When combing multiple *independent* and *diverse* decisions:
  - Each decision is more accurate than random guessing
  - Random errors cancel each other out
  - Correct decisions are reinforced
  - Classification accuracy increases

Different types of ensemble learning

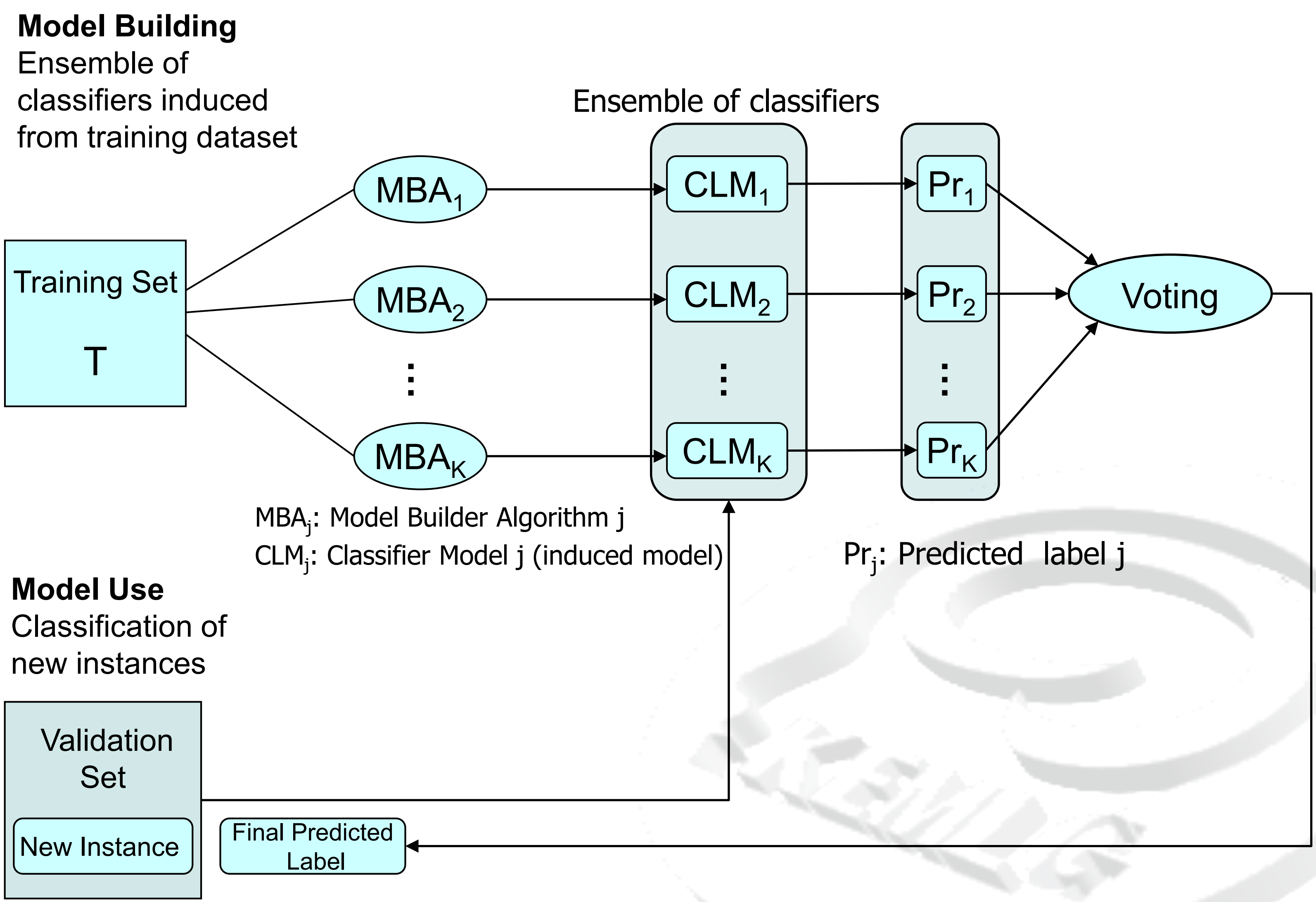
- Different learning **algorithms**
- Algorithms with different choice for **hyper-parameters**
- Data set with different **features** (e.g. random subspace)
- Data set = different **subsets** (e.g. bagging, boosting)

A possible classification

- Multiexpert combination methods (parallel classifier models):
  - Global approach (classifier models' fusion)
    - **Voting**: voting among different classifiers
    - **Bagging**: resample training data (bootstrapping), same classifier and voting
    - **Stacking**: a combiner learner (meta-learner), which combines the predictions of the classifiers
    - **Randomizing input features**: random subsets of features at each node (random forests)
  - Local approach (classifier models' selection)
    - **Gating**: Meta selection of the best classifier/s (best local expert/s) to be used (*Mixture of experts ensemble*)
- Multistage combination (sequential classifier models)
  - **Boosting**: Reweight training data. Next Learner focusing on misclassified instances by previous classifier
  - **Cascading**: Increasing complexity of learners
- Decorating methods: Adding artificial training data (noise addition)

Voting

- Different algorithms, same set of training data



Bagging (Bootstrap Agreggating) (1)

- Bootstrap aggregating. Create ensembles by repeatedly randomly resampling the training data [Breiman, 1996].

- Given a training set of size *n*, create *m* samples of size *n* by drawing *n* examples from the original data, *with replacement*.
  - Each ***bootstrap sample*** will on average contain 63.2% of the unique training examples, the rest are replicates.

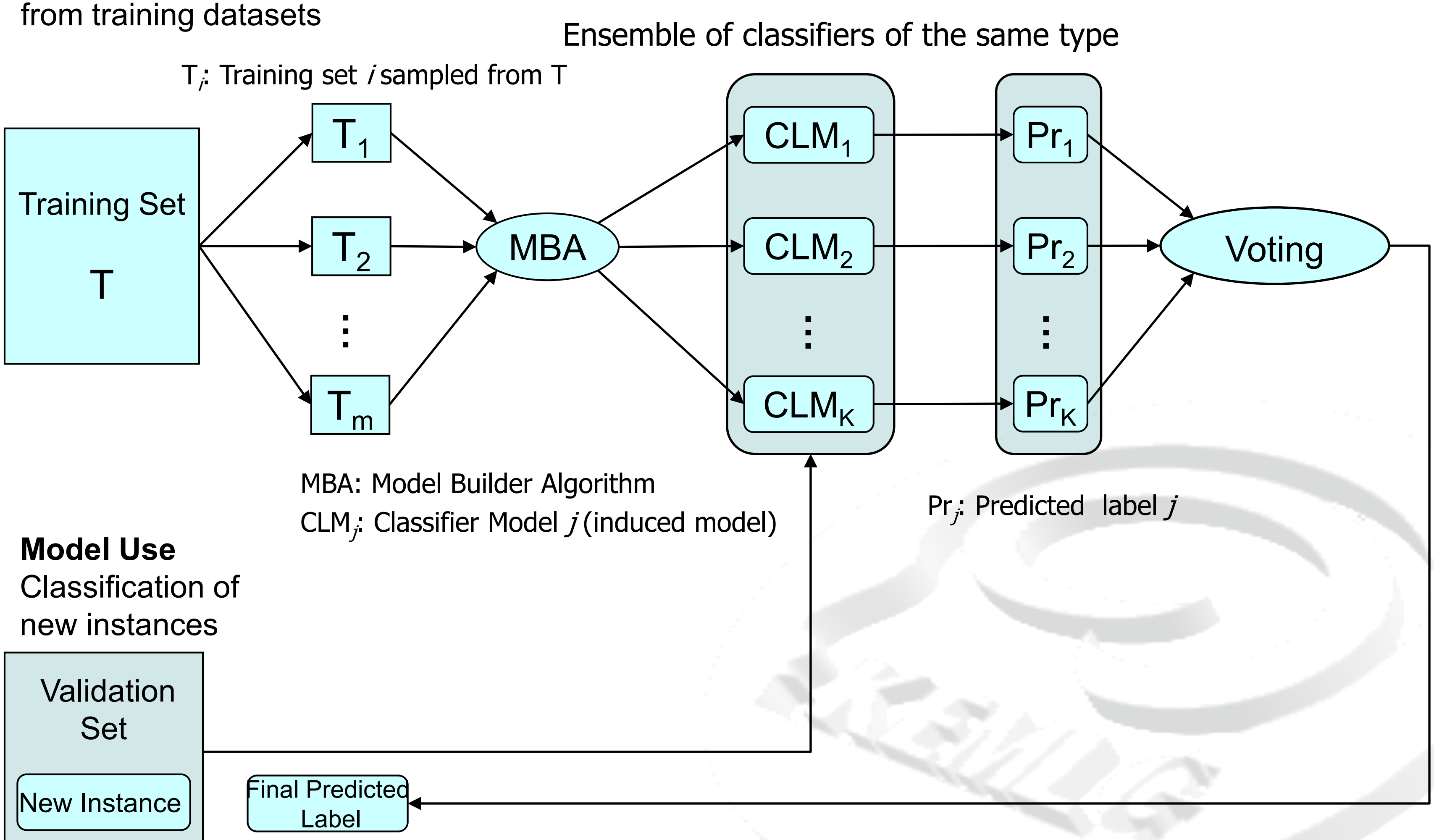
- Combine the *m* resulting models using simple majority vote.

- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.

Bagging (Bootstrap Agreggating) (2)

- Same algorithm, different versions of training dataset:

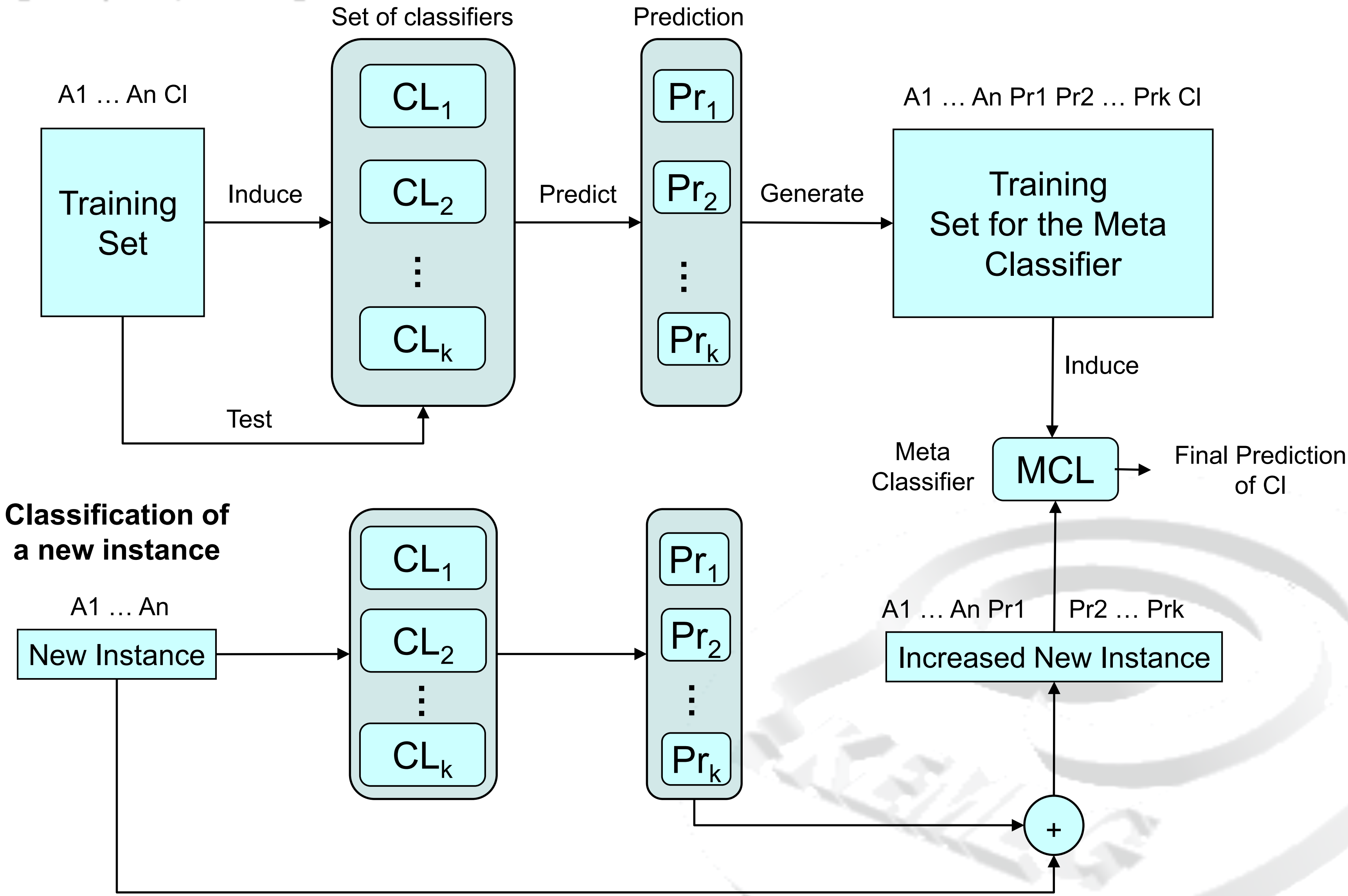
Model Building
Ensemble of classifiers induced from training datasets



Model Use
Classification of new instances

Stacking (Stacked Generalization)

[Wolpert, 1992]



Random Decision Forests / Random Forests

[Ho, 1998] / [Breiman, 2001]

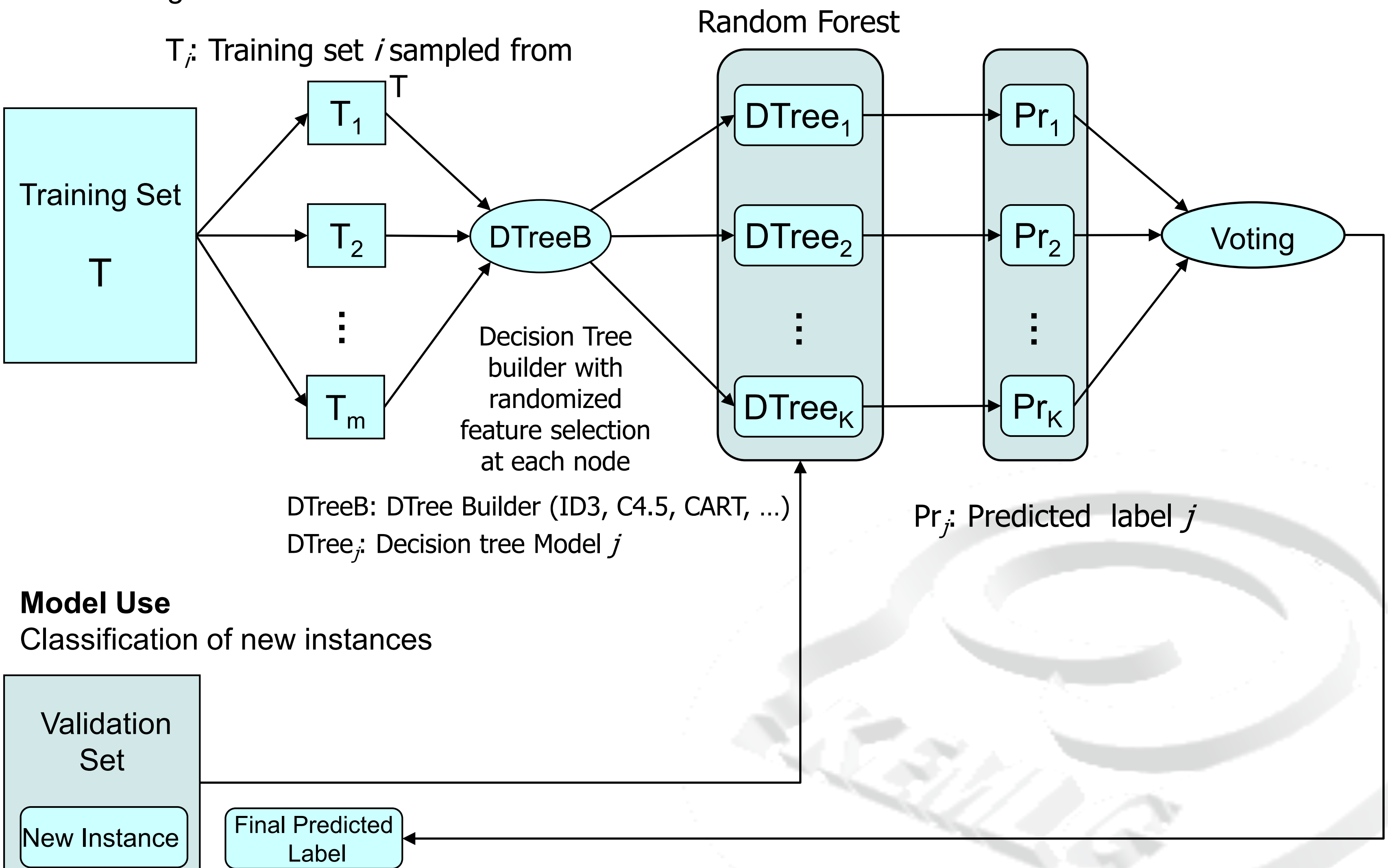
- References:
  - Tin Kam Ho. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8):832-844, 1998.
  - Leo Breiman. Random Forests. *Machine Learning*, 45:5-32, 2001

- Motivation: reduce error correlation between classifiers
- Main idea: build a larger number of un-pruned decision trees
- Ho's proposal: each tree is grown using a random subspace (selection) of features, which is the same for all the node splits
- Breiman's proposal: each tree uses a random subspace (selection) of features to split on at each node, and the training set for each tree is sampled (bootstrapping) from the original dataset

Random Forests

[Breiman, 2001]

Model Building
Ensemble of classifiers induced from training datasets



Model Use
Classification of new instances

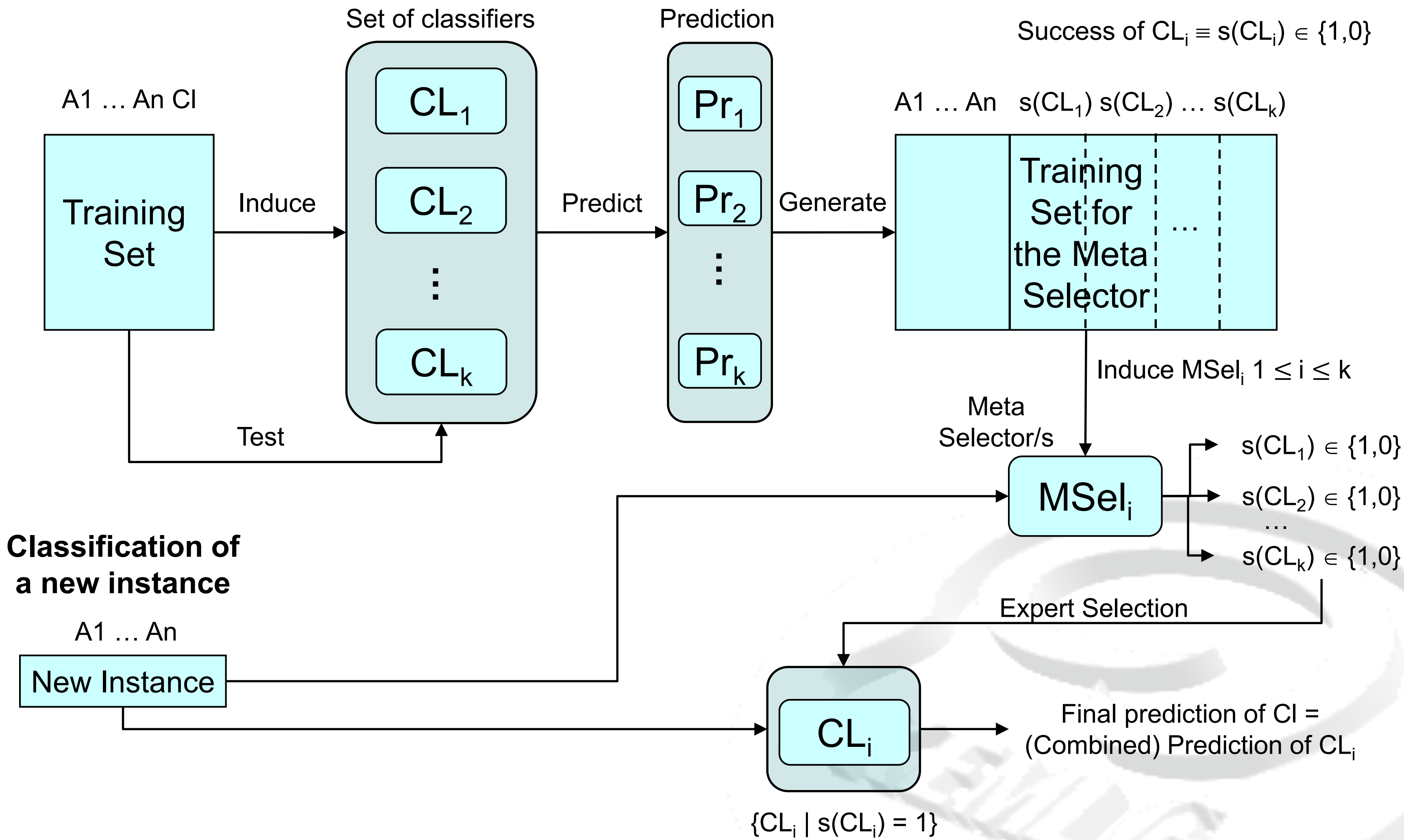
How Random Forests Work

[Breiman, 2001]

- Each tree is grown on a bootstrap sample of the training set of **N** cases.
- A number **F** is specified much smaller than the total number of variables **M:**
  - F = sqrt (M) or
  - F = int ($\log_2$ M +1))
- At each node, *F* variables are selected at random out of the M.
- The split used is the best split on these **F** variables according to the decision tree strategy.
- Final classification is done by majority vote across trees.

Gating (Mixture of Experts/Experts' Selection)

[Jacobs et al., 1991]



Boosting (1)
[Schapire, 1990]

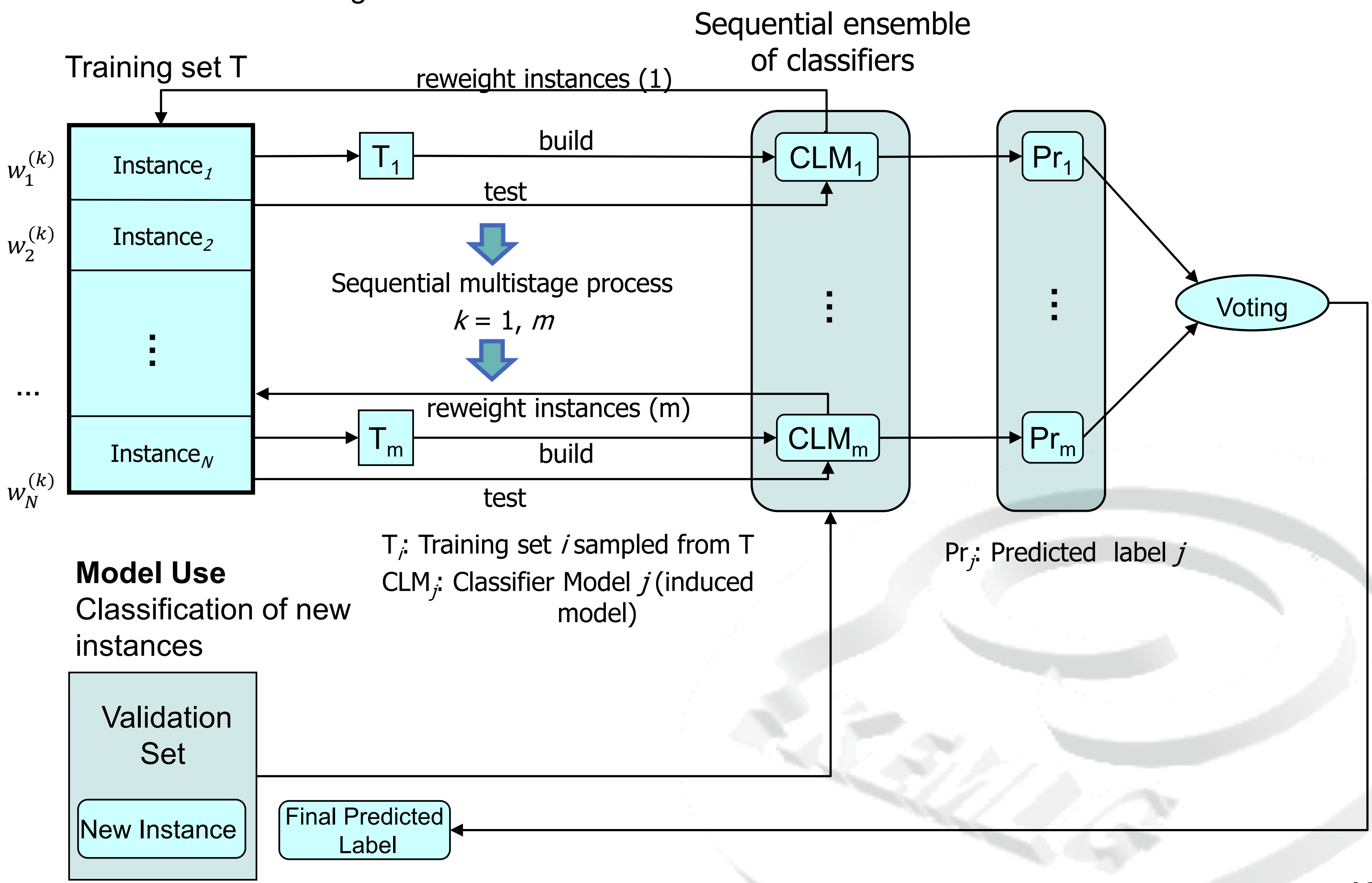
- Originally developed by computational learning theorists to guarantee *performance improvements* on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 [Schapire, 1990].

- Revised to be a practical algorithm, **AdaBoost**, for building ensembles that empirically improves generalization performance [Freund & Shapire, 1996].

- Examples are given weights. At each iteration, a new hypothesis is learned and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.

Boosting (2)

Model Building
Ensemble of classifiers sequentially induced from training datasets



Model Use
Classification of new instances

Boosting: basic algorithm

- General boosting algorithm:

  Set all examples to have equal uniform weights
  for *t* from 1 to *T* **do**
    Learn a hypothesis/model, h_t from the weighted examples
    Decrease the weights of examples h_t classifies correctly
  endfor

- Base (weak) learner must **focus on** *correctly classifying the most highly weighted examples* while strongly avoiding over-fitting.

- During *testing*, each of the *T* hypotheses/models get a weighted vote proportional to their accuracy on the training data.

AdaBoost

[Freund & Shapire, 1996]

TrainAdaBoost(D, BaseLearn)
for each example d_i in D **do**
let its weight w_i=1/|D|
endforeach
Let H be an empty set of hypotheses
for t from 1 to T **do**
Learn a hypothesis, h_t, from the weighted examples: h_t=BaseLearn(D)
Add h_t to H
Calculate the error, ε_t, of the hypothesis h_t as the total sum weight of the examples that it classifies incorrectly
If $\varepsilon_t > 0.5$ **then** exit loop, **else** continue
Let $\beta_t = \varepsilon_t \,/\, (1 - \varepsilon_t)$
Multiply the weights of the examples that h_t classifies correctly by β_t
Rescale the weights of all of the examples so the total sum weight remains 1.
endfor
return H

TestAdaBoost(ex, H)
Let each hypothesis, h_t, in H vote for ex's classification with weight $\log(1/\beta_t)$
return the class with the highest weighted vote in total.

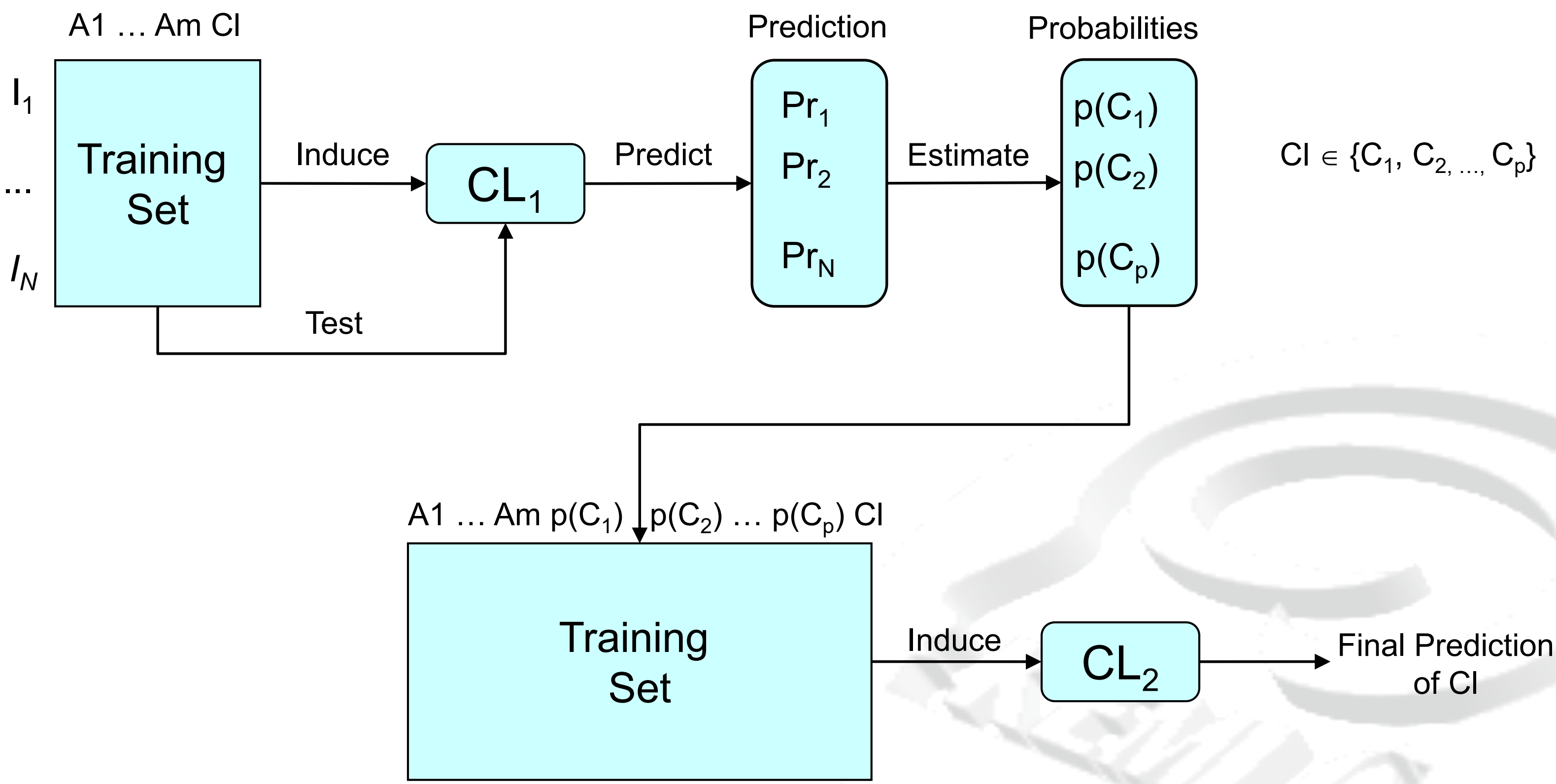
Learning with Weighted Examples

- Generic approach is to *replicate examples in the training set proportional to their weights*
  - $\frac{w_i}{1=\sum w_i} = \frac{n_i}{N}$, *thus* $n_i = w_i * N$
  - e.g., if we have a total number of N=1000 examples, in the weighted sample there should be:
    - 10 replicates of an example with a weight of 0.01
    - 100 replicates of one example with weight 0.1
- Most algorithms can be enhanced to efficiently incorporate weights directly in the learning algorithm so that the effect is the same (e.g. implement the WeightedInstancesHandler interface in WEKA).
- For decision trees, for calculating information gain, when counting example *i*, simply increment the corresponding count by w_i rather than by 1.

Cascading

[Viola & Jones, 2001]

- Sequence of several classifiers, using all information collected from the output from a previous classifier as additional information for the next classifier in the *cascade*



Experimental Results on Ensembles

[Freund & Schapire, 1996; Quinlan, 1996]

- Ensembles have been used to *improve generalization accuracy* on a wide variety of problems.
- On average, *Boosting* provides a larger increase in accuracy than *Bagging*.
- *Boosting* on rare occasions can degrade accuracy.
- *Bagging* more consistently provides a modest improvement.
- *Boosting* is particularly subject to over-fitting when there is significant noise in the training data.
- *Bagging* is easily parallelized.
- *Boosting* is not easily parallelized.

Random Forests vs Adaboost

- Error rates compare favorably to Adaboost
- More robust with respect to noise.
- More efficient on large data
- Provides an estimation of the importance of features in determining classification

Ensemble Methods

- RapidMiner operators:
  - Modeling/Predictive/Ensembles:
    - Bagging
    - Adaboost
    - Vote (different classifiers)
    - Stacking (different classifiers training a high level classifier)
    - …
  - Modeling/Predictive/Trees:
    - Random Forest
- Python
  - Scikit Learning
    - BaggingClassifier
    - RandomForestClassifier
    - ….
- R
  - Caret package

Intelligent Data Science and Artificial Intelligence (IDEAI-UPC)

Miquel Sànchez-Marrè
miquel@cs.upc.edu

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

https://kemlg.upc.edu