Knowledge Discovery from Data Streams: Multiple Models

João Gama

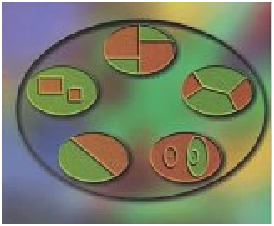
LIAAD-INESC Porto, University of Porto, Portugal
jgama@fep.up.pt

Outline

Multiple Models

Different learning algorithms exploit:

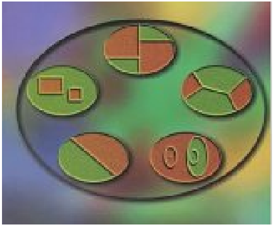
- Different languages for representing generalizations of the examples;
- Different search spaces;
- Different evaluation functions of the hypothesis;

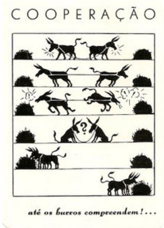


Multiple Models

How to take advantage of these differences?

Would be possible to obtain an ensemble of classifiers with a performance better than each individual classifier?





Observation: There is no overall better algorithm.

- Experimental results from Statlog and Metal project;
- Theoretical Results: *No free lunch* theorems.

A necessary condition

A necessary condition

An ensemble of classifiers improve over individual classifiers iif they disagree. *Hansen & Salamon - 1990*

How to measure the degree of disagreement?

The Error Correlation Metric

The Error Correlation Metric

Probability that two classifiers make the same error given that one of them is in error.

$$\phi_{i,j}(x) = P(\hat{f}_i(x) = \hat{f}_j(x) | \hat{f}_i(x) \neq f(x) \vee \hat{f}_i(x) \neq f(x))$$

A	1	1	1	1	1	1	0	1
B	0	1	1	0	1	1	0	0
Y	0	0	0	1	0	1	1	0

$$\phi_{A,B} = 4/7$$

The Error Correlation Metric

- Measures the diversity between the predictions of two algorithms;
- High values of ϕ: low diversity, redundant classifiers: the same type of errors
- Low Values of ϕ: high diversity: different errors.

Is the correlated error a sufficient condition?

Multiple Models

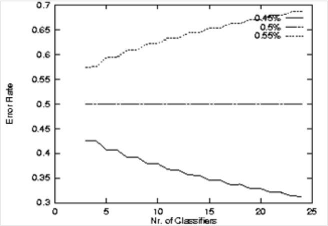
A simulation study:

- Consider a decision problem with two equi-probable classes: $P(Class_1) = P(Class_2)$
- The number of classifiers in the ensemble varies between [3, ..., 25].
- All classifiers have the same probability of error. Assume $P_{error}(Classifier_i) = \{0.45; 0.5; 0.55\}$

Multiple Model: aggregate the predictions of individual classifiers

- For each example
  - Each classifier predict a class label.
  - Count the votes for each class
  - Predict the most voted class: *uniform voting*.

Multiple Models: Simulation



Study how the error varies when varying the number of classifiers in the ensemble.
Probability of error of each classifier:

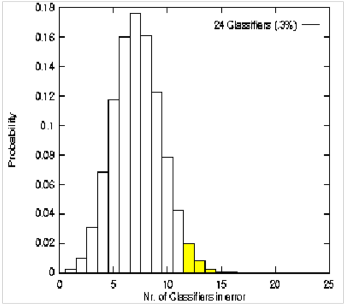
- $P = 0.5$ (random choice)
  The error of the ensemble is constant: 0.5
- $P > 0.5$
  The error of the ensemble increases linearly with the number of classifiers.
- $P < 0.5$
  The error of the ensemble **decreases** linearly with the number of classifiers.

Another Necessary Condition

Necessary Condition

The error of the ensemble decreases, with respect to each individual classifier, iif each individual classifier has a performance better than a random choice.

Multiple Models: Simulation



Assume an ensemble of 23 classifiers:

- probability of error of each classifier: 30%;
- aggregation by uniform vote.

Given a test example:

- the ensemble will be in error iif 12 or more classifiers are in error.
- The probability of error in the ensemble is given by the area under the curve of a binomial distribution;
- In this case this area is 0.026.
- Much less than each individual classifier

Necessary Conditions

To achieve higher accuracy the models should be diverse and each model must be quite accurate Ali & Pazzani 96

Necessary Conditions

Classifiers in the ensemble, should have:

- performance better than random guess;
- non-correlated errors;
- errors in different regions of the instance space.

Outline

Weighted-Majority Algorithm

Littlestone and Warmuth. *The weighted majority algorithm*. Information and Computation, 108(2):212-261, 1994.

- Makes predictions by taking a weighted vote among a pool of predictors
- Learn by changing the weights associated with each learning algorithm.
- The only thing required for the learning algorithm is that it makes a prediction
  - Can be single attribute

Weighted-Majority Algorithm

- The Algorithm:

Initial conditions: For all $i \in \{1, \ldots, M\}$, $w_i = 1$.

Weighted-Majority(x)

For $i = 1, \ldots, M$,

$y_i = h_i(x)$.

If $\sum_{i:y_i=1} w_i \geq \sum_{i:y_i=0} w_i$, then return 1, else return 0.

If the target output y is not available, then exit.

For $i = 1, \ldots, M$,

If $y_i \neq y$ then $w_i \leftarrow \frac{w_i}{2}$.

- Classify a test example using weighted vote among predictions of each expert

Weighted-Majority Algorithm

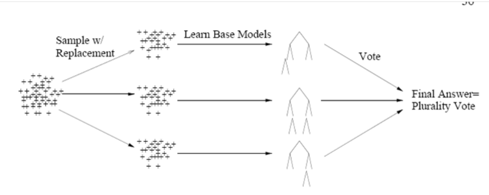
- Interesting fact:
  - Let D be any sequence of examples
  - Let A be a set of n predictors
  - Let k be the minimum number of mistakes made by any algorithm in A for the set of examples D

  - The number of mistakes made by the weighted-majority algorithm using β=1/2, is *at most*: $2.4 \times (k + log_2 n)$

Outline

Bagging

- Bagging (Breiman 1996):
  - Given a Training set
    - Generate multiple samples with replacement
    - For each sample generate a decision model
  - Given a Test example
    - Each base model makes a prediction
    - Combine predictions by uniform voting



Bagging

- Main Characteristics
  - Require Unstable Algorithms.
    - Algorithms sensible to small perturbations on the training set - Decision Trees, Neural Nets
    - Easy to implement as a wrapper over any learning algorithm
  - Easy for parallel environments
- Variance reduction method

Online Bagging

- Bagging seems to require that the entire training set be available at all times
  - for each base model, sampling with replacement is done by performing random draws over the entire training set.
- Each original training example may be replicated zero, one, two, or more times in each bootstrap training set
  - the sampling is done with replacement.
- Each base model's bootstrap training set contains K copies of each of the original training examples where P(K=k) is given by a binomial distribution

Online Bagging

Oza, **Online Ensemble Learning**, PhD thesis, University of California, Berkeley, 2001

- A binomial distribution: the probability of obtaining exactly k successes in N trials.
- As N increases the distribution of k tends to a Poisson(1) distribution.
  - The Poisson distribution is the limit of a binomial when N tends to infinity.

$$P(K = k) = \frac{\lambda^k exp(-\lambda)}{k!}$$

  - λ=1 corresponds to assign a weight = 1/N to each example

Online Bagging

- As N increases the distribution of k tends to a Poisson(1) distribution:

$$P(K = k) = \frac{exp(-1)}{k!}$$

```
OnlineBagging(h, d)
  For each base model h_m, (m ∈ {1, 2, ..., M}) in h,
      Set k according to Poisson(1).
      Do k times
          h_m = L_o(h_m, d)
  Return h(x) = argmax_{y∈Y} Σ_{m=1}^{M} I(h_m(x) = y).
```

Figure 3.3: Online Bagging Algorithm: **h** is the classification function returned by online bagging, d is the latest training example to arrive, and L_o is the online base model learning algorithm.

Boosting

- An iterative (sequential) algorithm
  - Each example as an weigh associated with.
  - The set of weights define a distribution over the training set
- Main Idea:
  - Initialize weights with a uniform distribution.
  - Iterate:
    - Generate a classifier from the actual distribution.
    - Increase the weights of misclassified examples.
    - Decrease the weights of correct classified examples.
  - All the classifiers are used to classify test examples by weighted voting.

Boosting

- Can be used in any learning algorithm,
    - The learning algorithm should be able to generate an hypothesis slight better than a random choice (*weak learner*).
    - Able to reduce *bias* and *variance*.

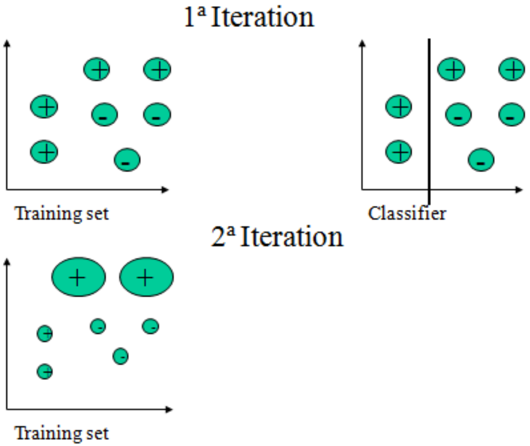
Boosting: Example

Weak learner – generate an hyper-plane perpendicular to one of the axis

1ª Iteration

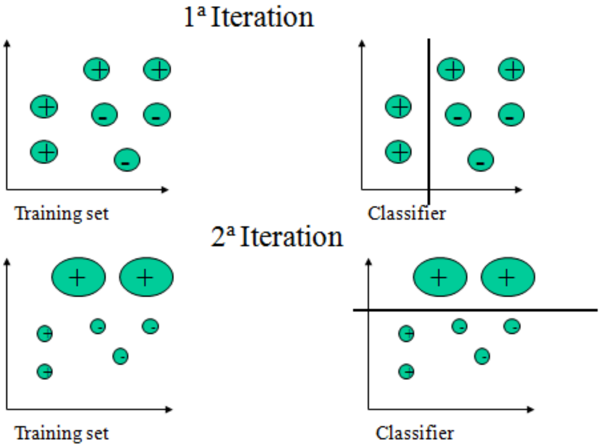
Boosting: Example

Weak learner – generate an hyper-plane perpendicular to one of the axis



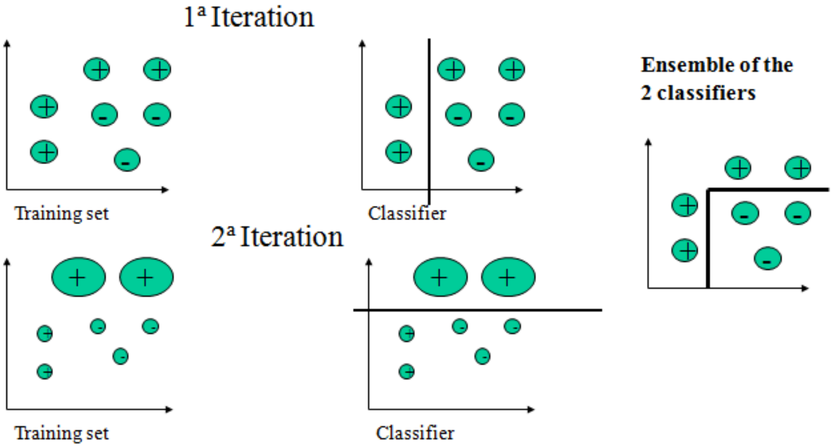
Boosting: Example

Weak learner – generate an hyper-plane perpendicular to one of the axis

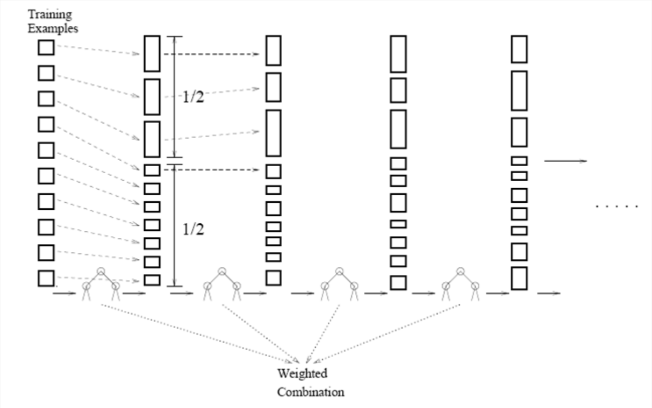


Boosting: Example

Weak learner – generate an hyper-plane perpendicular to one of the axis



Boosting

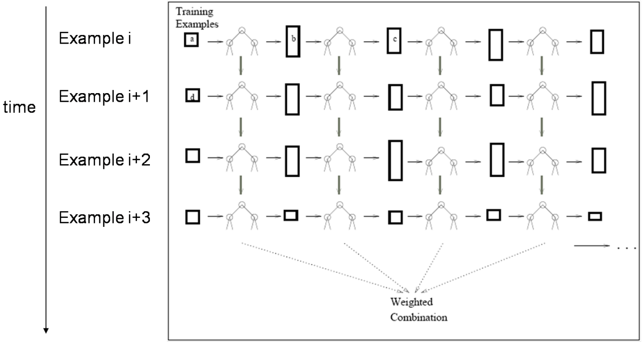


Online Boosting

Oza, **Online Ensemble Learning**, PhD thesis, University of California, Berkeley, 2001

- Like in Bagging the initial set of weights is uniform
  - $\lambda = 1$ (poisson(1))
  - For subsequent base models, online boosting updates the Poisson parameter for each training example in a manner very similar to the way batch boosting updates the weight of each training example:
    - increasing it if the example is misclassified
    - decreasing it if the example is correctly classified.

Online Boosting



The weight of rectangles indicates the weight of the examples.

Outline

Dual Perturb & Combine

Approaches that use only a single model and delays at the prediction stage the generation of multiple predictions by perturbing the attribute vector corresponding to a test case.

Dual Perturb & Combine

Geurts & Wehenkel *Closed-form dual perturb and combine for tree-based models*. In Proc. of the 22nd international Conference on Machine Learning, 2005

- Only a single model is generated from the training set.
- In the prediction phase, each test example is perturbed several times.
    - To perturb a test example, white noise is added to the attribute-values.
    - The predictive model makes a prediction for each perturbed version of the test example.
    - The final prediction is obtained by aggregating the different predictions.
- Geurts presents evidence that this method is efficient in variance reduction.

Outline

1. Motivation
2. Weighted-Majority Algorithm
3. Perturbing Training Examples
4. Perturbing Test Examples
5. Concept Drift and Multiple Models
6. An Application Example

Concept Drift and Multiple Models

- A common assumption in dynamic environments is that data can be generated by several distributions
  - At least in phases with concept drift.
- A natural approach to deal with drifting concepts is multiple models

Concept Drift and Multiple Models

Mining Concept Drifting Data Streams using Ensemble Classifiers; H. Wang, W. Fan, P. Yu, J. Han; KDD 2002

- Propose a general framework for **mining** concept-drifting **data streams** using weighted ensemble classifiers.
- They train an ensemble of classification models, such as C4.5, RIPPER, naive Bayesian, etc.,
  - from sequential chunks of the **data** stream.
  - The classifiers in the ensemble are weighted based on their expected classification accuracy on the test **data** under the time-evolving environment.

Concept Drift and Multiple Models

Kolter and Maloof, *Using additive expert ensembles to cope with Concept drift*, Proc. 22 International Conference on Machine Learning, 2005

- The Dynamic Weighted Majority (DWM) maintains an ensemble of base learners, predicts using a weighted-majority vote of these *experts*, and dynamically creates and deletes experts in response to changes in performance.
- DWM maintains an ensemble of predictive models, referred to as experts, each with an associated weight.

Concept Drift and Multiple Models

- Experts can use the same algorithm for training and prediction;
- They are created at different time steps so they use different training set of examples.
- The final prediction is obtained as a weighted vote of all the experts.
  - For each class, DWM sums the weights of all the experts predicting that class, and predicts the class with greatest weight.
- The learning element of DWM, first predicts the classification of the training example.
- The weights of all the experts that misclassified the example are decreased

AddExp for Classification

Algorithm AddExp.D($\{\mathbf{x}, y\}^T, \beta, \gamma$)

Parameters:

$\{\mathbf{x}, y\}^T$: training data with class $y \in Y$

$\beta \in [0, 1]$: factor for decreasing weights

$\gamma \in [0, 1]$: factor for new expert weight

Initialization:

1. Set the initial number of experts $N_1 = 1$.
2. Set the initial expert weight $w_{1,1} = 1$.

For $t = 1, 2, \ldots, T$:

1. Get expert predictions $\xi_{t,1}, \ldots, \xi_{t,N_t} \in Y$
2. Output prediction:
$$\hat{y}_t = \operatorname{argmax}_{c \in Y} \sum_{i=1}^{N_t} w_{t,i} [c = \xi_{t,i}]$$
3. Update expert weights:
$$w_{t+1,i} = w_{t,i} \beta^{[y_t \neq \xi_{t,i}]}$$
4. If $\hat{y}_t \neq y_t$ then add a new expert:
$$N_{t+1} = N_t + 1$$
$$w_{t+1,N_{t+1}} = \gamma \sum_{i=1}^{N_t} w_{t,i}$$
5. Train each expert on example $\mathbf{x}_t, y_t$.

Figure 1. **AddExp for discrete class predictions.**

AddExp for Regression

Algorithm AddExp.C($\{\mathbf{x}, y\}^T, \beta, \gamma, \tau$)

Parameters:

- $\{\mathbf{x}, y\}^T$: training data with class $y \in [0, 1]$
- $\beta \in [0, 1]$: factor for decreasing weights
- $\gamma \in [0, 1]$: factor for new expert weight
- $\tau \in [0, 1]$: loss required to add a new expert

Initialization:

1. Set the initial number of experts $N_1 = 1$.
2. Set the initial expert weight $w_{1,1} = 1$.

For $t = 1, 2, \ldots, T$:

1. Get expert predictions $\xi_{t,1}, \ldots, \xi_{t,N_t} \in [0, 1]$
2. Output prediction:
$$\hat{y}_t = \frac{\sum_{i=1}^{N_t} w_{t,i}\xi_{t,i}}{\sum_{i=1}^{N_t} w_{t,i}}$$
3. Suffer loss $|\hat{y}_t - y_t|$
4. Update expert weights:
$$w_{t+1,i} = w_{t,i}\beta^{|\xi_{t,i} - y_t|}$$
5. If $|\hat{y}_t - y_t| \geq \tau$ add a new expert:
$$N_{t+1} = N_t + 1$$
$$w_{t+1,N_{t+1}} = \gamma \sum_{i=1}^{N_t} w_{t,i}|\xi_{t,i} - y_t|$$
6. Train each expert on example $\mathbf{x}_t, y_t$.

Figure 2. **AddExp** for continuous class predictions.

AddExp for Regression

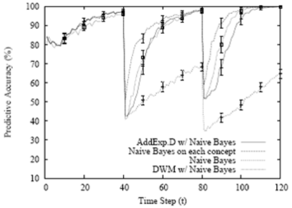


Figure 3. Predictive accuracy on the STAGGER concepts.

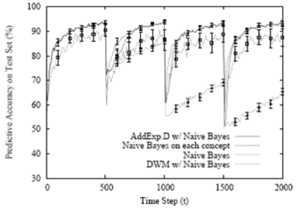


Figure 4. Predictive accuracy on the hyperplane problem.

Add Expert

- Add expert adds too many experts!
  - Add an expert only after observing a significant increase of the error.
- Pruning strategies:
  - Eliminate oldest experts.
  - Eliminate experts with high error in the most recent examples.

Concept Drift and Multiple Models

Combining Proactive and Reactive Predictions for Data Streams; Y. Yang, X. Wu and X. Zhu; KDD05

- Presents a system that:
  - Identifies new concepts
  - Identifies re-appearing concepts,
  - Learns transition patterns among concepts to help prediction.

Concept Drift and Multiple Models

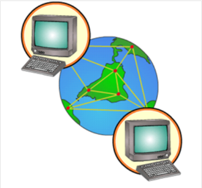
- Different from conventional methods that passively waits until the concept changes.
  - the system incorporates proactive and reactive predictions.
  - In a proactive mode, it anticipates what the new concept will be if a future concept change takes place,
  - prepares prediction strategies in advance.
- It uses a transition matrix between concepts.

Outline

Introduction

Data Streams:

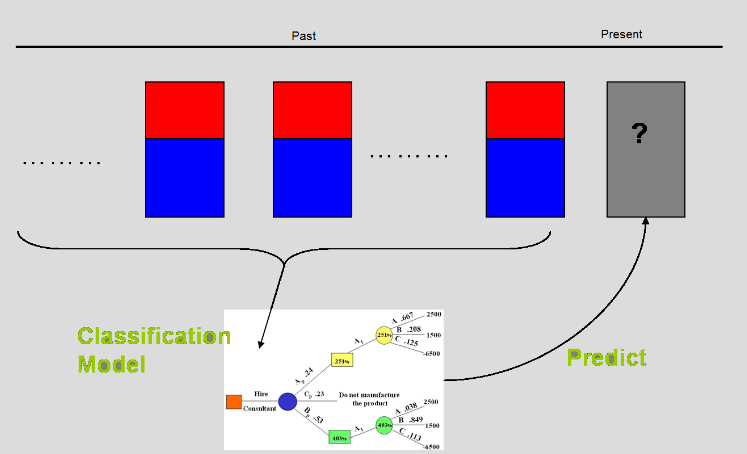
- Continuously arriving data flow;
- Applications: network traffic, credit card transaction flow, phone calling records, etc.



Stream Classification

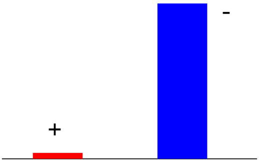
- Construct a classification model based on past records
- Use the model to predict labels for new data
- Help decision making

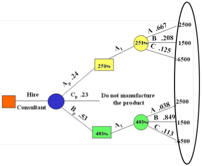
Framework


Past
Present
?
Classification Model
Predict
Hire
Consultant
Do not manufacture the product
2500
1500
6500

Mining Skewed Data Stream

- Skewed Distribution
  Credit card frauds, network intrusions
- Existing Stream Classification Algorithms used to be evaluated on balanced data
- Problems:
  Ignore minority examples
  The cost of misclassifying minority examples is usually huge

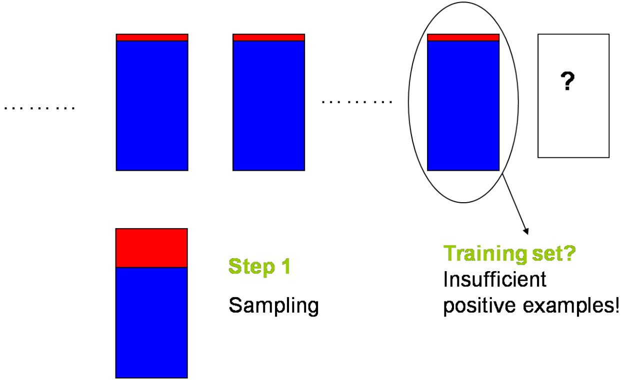




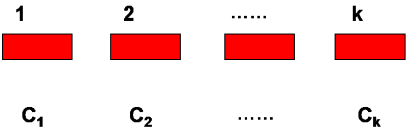
Classify every leaf node as negative

Stream Ensemble Approach

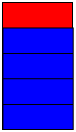
Learning from batches of examples over time:



Stream Ensemble Approach



$$f^E(x) = \frac{1}{k}\sum_{i=1}^{k} f^i(x)$$



Step 2

Ensemble

Analysis

- Incorporation of old positive examples
  increase the training size, reduce variance
  negative examples reflect current concepts, so the increase in boundary bias is small
- Ensemble
  reduce variance caused by single model
  disjoint sets of negative examples: the classifiers will make uncorrelated errors

Bibliography

- N. Littlestone and M. K. Warmuth. The weighted majority algorithm. Information and Computation, 108(2):212–261, 1994.
- Nikunj Oza, **Online Ensemble Learning**, PhD thesis, University of California, Berkeley, 2001
- Wang, W. Fan, P. Yu, J. Han; Mining Concept Drifting Data Streams using Ensemble Classifiers; KDD 2002
- Kolter, J.Z. and Maloof, M.A., Dynamic Weighted Majority: A new ensemble method for tracking concept drift, Proceedings of the Third IEEE International Conference on Data Mining, 2003
- J. Kolter and M. Maloof, Using additive expert ensembles to cope with Concept drift, Proceedings of the Twenty-second International Conference on Machine Learning, 2005
- Y. Yang, X. Wu and X. Zhu; Combining Proactive and Reactive Predictions for Data Streams; KDD05, 2005