Learning in Dynamic Environments: Decision Trees for Data Streams Physiological Data Contest

João Gama, Pedro Rodrigues
LIACC
University of Porto

Motivation

- The physiological data contest:
  - Large amounts of sequential data
  - Sensor fusion
  - Hidden variables
- Our approach
  - Online Learn a predictive model from the data stream

Design Criteria for Learning from Data Streams

- Data-streams
  - Open-ended data flow
  - Continuous flow of data
- Data Mining on Data streams:
  - Processing each example
    - Small constant time
    - Fixed amount of main memory
  - Single scan of the data
    - Processing examples at the speed they arrive
  - Classifiers at *anytime*
    - Ideally, produce a model equivalent to the one that would be obtained by a batch data-mining algorithm
  - The data-generating phenomenon could change over time
    - Concept drift

Related Work

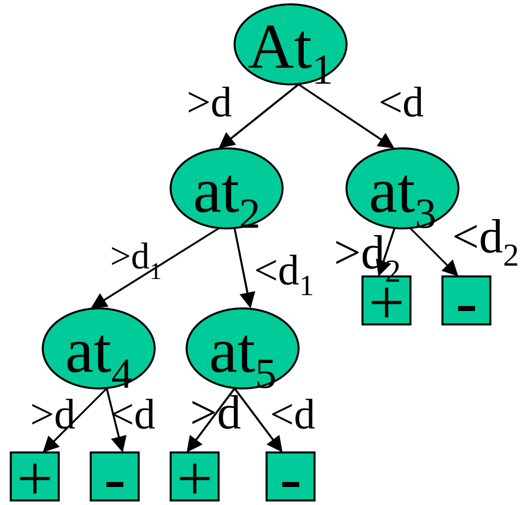
- Incremental Trees
  - Decision Trees for Data streams
    - ➢ Very Fast Decision Trees for Mining High-Speed Data Streams (P. Domingos, et al., KDD 2000)
      - When should a leaf become a decision node?
        - » Hoeffding Bound
      - Nominal Attributes
  - VFDTc (Gama, R.Rocha, P.Medas, KDD03)
    - Numerical attributes
    - Functional leaves
- Non-Incremental Trees
  - Functional Leaves
    - Perceptron Trees (P.Utgoff, 1988)
    - Nbtree (R. Kohavi, KDD 96)
  - Splitting Criteria
    - Split Selection Methods For Classification Tress (W. Loh, Y. Shih, 1997)
      - Two-class problems

Ultra-Fast Forest of Trees

- Main characteristics:
  - Incremental, works online
  - Continuous attributes
  - Single scan over the training data
    - Processing each example in constant time
  - Forest of Trees
    - A *n* class-problem is decomposed into *n*(n-1)/2* two-classes problem
    - For each binary problem generate a decision tree
  - Functional Leaves
    - Whenever a test example reach a leaf, it is classified using
      - The majority class of the training examples that fall at this leaf.
      - A naïve Bayes built using the training examples that fall at this leaf.
      - A IDBD classifier built using the training examples that fall at this leaf.
    - Anytime classifier

Binary decision trees for data streams

- Growing a single tree
  - Start with an empty leaf
  - While TRUE
    - Read next example
    - Propagate the example through the tree
      - From the root till a leaf
    - For each attribute
      - Update sufficient statistics
        - » Statistics to compute *mean* and *standard deviation*
        - » Nx, Sx, Sx2
    - Estimate the gain of splitting
      - For each attribute
        - » Compute the cut-point given by quadratic discriminant analysis
        - » Estimate the information gain
      - If the Hoeffding bound between the two best attributes is verified
        - » The leaf becomes a decision node with two descendent leaves

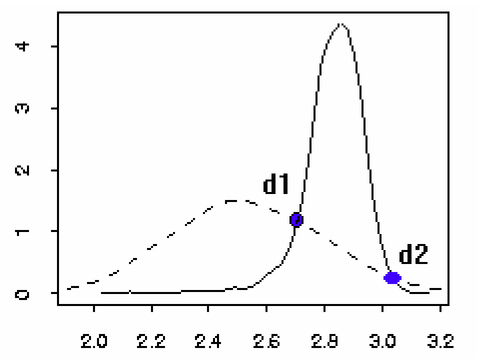


The splitting criteria

- The case of two classes.
- All candidate splits will have the form of $\text{Attribute}_i <= \text{value}_j$
  – For each attribute, quadratic discriminant analysis defines the cut-point.
  – Assume that for each class the attribute-values follows a ***univariate*** normal distribution
    – N(mean, standard deviation).
    – Where p(i) is the probability that an example that fall at leaf *t* is from classe I
  - The best cut-point is the solution of: $p(+)N(\bar{x}_+, \sigma_+) = p(-)N(\bar{x}_-, \sigma_-)$
    – A quadratic equation with at most two solutions: d1, d2
  - The solutions of the equation split the X-axis into three intervals:

$$(-\infty; d1); (d1, d2); (d2; +\infty)$$

  – We choose between d1 or d2, the one that is closer to the sample means.



Estimating the gain of a cut-point

	Att_i<=d	Att_i>d
Class+	$\mathbf{p_1^+}$	$\mathbf{P_2^+}$
Class -	$\mathbf{p_1^-}$	$\mathbf{P_2^-}$

- For each Attribute
  - The cut point defines a contingency table.
  - The information gain is:

$$G(Att_i) = \text{info}(p^+, p^-) - \sum_j (p_j * \text{info}(p_j^+, p_j^-))$$

where

$$\text{info}(p^+, p^-) = -p^+ \log_2 p^+ - p^- \log_2 p^-$$

- The attributes are sorted by information gain.
  - $G(X_a)>G(X_b)> \ldots > G(X_c)$
- When should we transform a leaf into a decision node?
  - When there is a high probability that the selected attribute is the wright one !

The Hoeffding bound

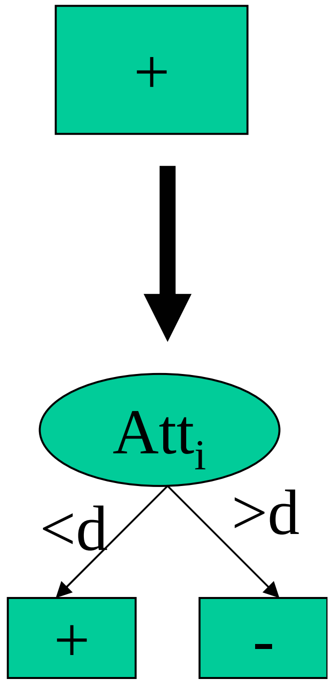
- Suppose we have made **n** independent observations of a random variable **r** whose range is **R.**
- The Hoeffding bound states that:
  - With probability 1-δ
  - The true mean of **r** is at least $\bar{r} \pm \varepsilon$ where $\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$
  - Independent of the probability distribution generating the examples.

- The heuristic used to choose test attributes is the information gain G(.)
  - Select the attribute that maximizes the information gain.
  - The range of information gain is log (#classes)
- Suppose that after seeing **n** examples, $G(X_a)>G(X_b)> \ldots > G(X_c)$
- Given a desired δ, the Hoeffding bound ensures that Xa is the correct choice if G(Xa)-G(Xb) > ε.
  - with probability 1- δ

From a leaf to a decision node

- The tree is expanded:
  - When the difference of gains between the two best attributes satisfies the Hoeffding bound,
    - A splitting test based on the best attribute is installed in the leaf
    - The leaf becomes a decision node with two descendent branches
  - When two or more attributes have very similar gains
    - Even given a large number of examples, and
    - The Hoeffding bound declares a *tie*.
      - Example: there are duplicate attributes.
    - The leaf becomes a decision node, if $\nabla G < \varepsilon < \tau$

      where τ is a user defined constant.



- How many examples should be required to trigger the evaluation of the splitting decision criteria?

$$n_{\min} = 1/(2 * \delta) * \log(2/\varepsilon)$$

Short Term Memory

- We maintain a limited number of the most recent examples.
- They are maintained on a *double queue*, that supports
  - Constant time for insertion of elements at the beginning of the sequence.
  - Constant time for deletion of elements at the end of the sequence.

- When the tree is expanded, two new leaves are generated.
  - The sufficient statistics of these new leaves are initialized with the examples at the short term memory.

Classification strategies at Leaves

- To classify a test example
  - The example traverses the tree from the root to a leaf,
    - Following the path given by the attribute values.
  - The leaf classifies the example.
- The usual strategy:
  - The test example is classified with the majority class from the training examples that reached the leaf.
  - In incremental learning, that
    - Maintain a set of sufficient statistics at each leaf
    - Only install a split test when there is evidence enough
    - More appropriate and powerful techniques should be applied!
  - We have implemented two other classification strategies:
    - Naive Bayes
    - Incremental Delta-Bar-Delta rule

Functional Leaves: Naïve Bayes

- Naive Bayes
  - Based on Bayes Theorem
    - Assuming the independence of the attributes given the class label
    - We assume that, for each class, the attribute-values follow a normal distribution
      - From the sufficient statistics stored at each leaf.
  - Naturally Incremental

  - A test example is classified in the class that maximizes:

$$P(Cl_i \mid \vec{x}) \propto \log(P(Cl_i)) + \sum_j \log(\phi(\bar{x}_k^i, \sigma_k^i))$$

Forest of Trees

- A multi-class problem is decomposed into a set of two-class problems.
  - A n class problem is decomposed into n(n-1)/2 binary problems.
    - A two-class problem for each possible pair of classes..
  - For each problem generate a decision tree
    - Leading to a forest of decision trees.
- Fusion of classifiers
  - To classify a test example:
    - Each decision tree classifies the example
      - Output a probability class distribution
    - The outputs of all decision trees are aggregated using the sum rule.

Experimental Evaluation: Physiological Data

- Tasks
  1. Predict the gender for every sessionId
  2. Identify when a person is participating in context 1
  3. Identify when a person is participating in context 2.
- The Data
  - For all tasks we have used as attributes:
    - Characteristics 1 and 2
    - Sensor 1-9
  - We have considered all the tasks as two-class problems

Task 1 Evaluation on training set

- Evaluation method:
  - Split the labelled set into two sets
    - Training set: 500000 records
    - Evaluation set: last 80264 records
  - Some points:
    - All users consistently classified
    - Confusion Matrix:
    - Training Time: 39 seconds

Task2 and 3: Evaluation on Training Set

- Skew class distribution
  - Consider misclassification costs

	N	P
N	0	10
P	0.1	0

  - Sequences on the training set:

Task2	*Nr.seq*	*Mean(Size)*	*Min(Size)*	*Max(Size)*
Prediction	2992	12.7	3	154
Observed	75	57	6	177
Task3	*Nr.seq*	*Mean(Size)*	*Min(Size)*	*Max(Size)*
Prediction	795	30.3	3	516
Observed	37	301	65	592

Conclusions

- Our solution:
  - Single model
  - Incremental and online model
    - Can incorporate new information
  - Fast training
  - Misclassification costs
  - Any time classifier

Thanks for your attention!

More information:

http://www.liacc.up.pt/~jgama