Learning Forests of Trees from Data Streams

João Gama, Pedro Medas, Pedro Rodrigues
LIACC
University of Porto

Adaptive Learning Systems – ALES
Project sponsored by Fundação Ciência e Tecnologia
Contract: POSI/SRI/39770/2001

Overview

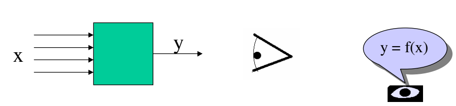
- Motivation
- Related Work
- Ultra-Fast Forest Trees
  - Binary Decision trees
    - Splitting Criteria
      - From Leaf to Decision Node
    - Functional Leaves
    - Functional Nodes
  - Forest of Trees
  - **Concept Drift**
- Experimental Work
  - Stationary Datasets
    - Sensitivity Analysis
  - Non-stationary Datasets
    - Electricity Market
- Conclusions

Aprendizagem Automática

- Áreas disciplinares
  - Estatística
    - Inferência estatística
  - Computação
    - Inteligência Artificial
      - Aprendizagem Automática
  - Bases de dados
    - Bases de Dados Multidimensionais
- Definições:
  - "*Self-constructing or self-modifying representations of what is being experienced for possible future use*" Michalski, 1990
  - "*Analysis of observational data to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful for the data owner*" Hand, Mannila, Smyth, 2001
  - Obter representações em compreensão a partir de representações em extensão.

Aplicações

- Códigos Postais
- Predição do uso da terra
- Aprender a conduzir veículos autónomos
- Web sites Adaptativos.



Códigos Postais (OCR)

Predição do Uso da Terra

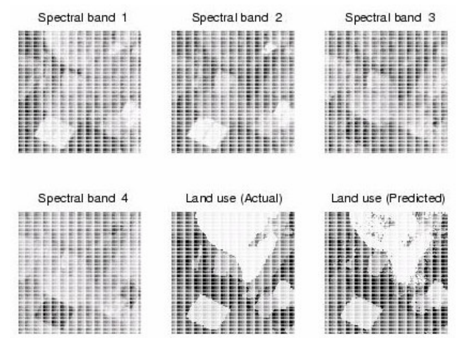
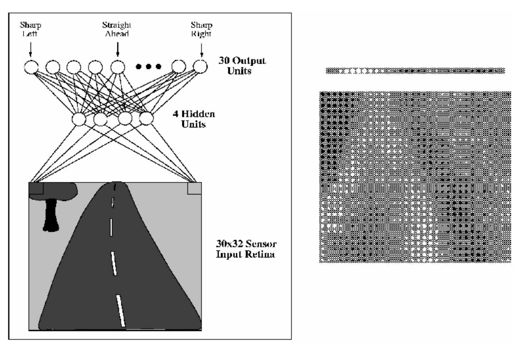


Fig. 9.3: Satellite image dataset. Spectral band intensities as seen from a satellite for a small (8.2*6.6 km) region of Australia. Also given are the actual land use as determined by on-site visit and the estimated classes as given by linear discriminants.

Veículos Autónomos

Veículos Autónomos



Web sites adaptativos

Diagnóstico á distância

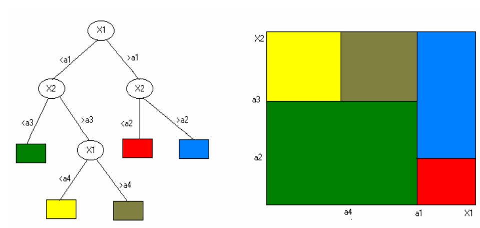
•Sensores:
- •gsr_low_average
- •heat_flux_high_average
- •near_body_temp_average
- •pedometer
- •skin_temp_average
- •longitudinal_accelerometer_SAD
- •longitudinal_accelerometer_average
- •transverse_accelerometer_SAD
- •transverse_accelerometer_average

The SenseWear armband, shown in the figure below, is a sleek, wireless and accurate wearable body monitor that enables continuous physiological monitoring outside the laboratory.

Árvores de Decisão

- Uma árvore de decisão utiliza uma estratégia de *dividir-para-conquistar*:
  - Um problema complexo é decomposto em sub-problemas mais simples.
  - Recursivamente a mesma estratégia é aplicada a cada sub-problema.
- A capacidade de discriminação de uma árvore vem da:
  - Divisão do espaço definido pelos atributos em sub-espaços.
  - A cada sub-espaço é associada uma classe.
- Crescente interesse
  - CART (Breiman, Friedman, et.al.)
  - C4.5 (Quinlan)
  - S_{plus}, Statistica, SPSS

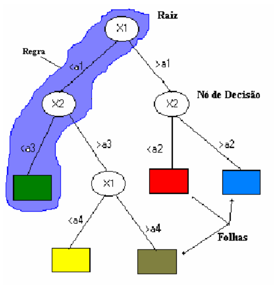
Árvores de decisão – Exemplo da partição do espaço dos atributos



$$y = \begin{cases} + \text{ if } at1 < a1 \text{ and } at2 > a3 \\ + \text{ if } \ldots \\ - \text{ if } at1 < 0.1 \text{ and } at2 > a3 \\ \ldots \end{cases}$$

Disjuntive Normal Form (DNF)

O que é uma Arvore de Decisão?



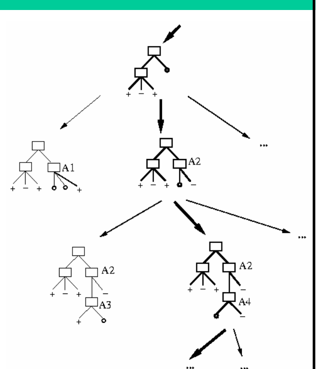
- Representação por árvores de decisão:
  - Cada nó de decisão contem um teste num atributo.
  - Cada ramo descendente corresponde a um possível valor deste atributo.
  - Cada Folha está associada a uma classe.
  - Cada percurso na arvore (da raiz à folha) corresponde a uma regra de classificação.
- No espaço definido pelos atributos:
  - Cada folha corresponde a uma região
    - Hiper-rectângulo
  - A intersecção dos hiper-rectângulos é vazio
  - A união dos hiper-rectângulos é o espaço completa.

Vantagens das Arvores de decisão

- Método não-paramétrico
  - Não assume nenhuma distribuição particular para os dados.
  - Pode construir modelos para qualquer função desde que o numero de exemplos de treino seja suficiente.
- A estrutura da árvore de decisão é independente da escala das variáveis.
  - Transformações monótonas das variáveis (log x, 2*x, ...) não alteram a estrutura da arvore.
- Elevado grau de interpretabilidade
  - Uma decisão complexa (prever o valor da classe) é decomposto numa sucessão de decisões elementares.
- É eficiente na construção de modelos:
  - Complexidade média O(n log n)
- Robusto á presença de pontos extremos e atributos redundantes ou irrelevantes.
  - Mecanismo de selecção de atributos.
- Comportamento no Limite: $erro(árvore)_{n \to \infty} = erro_{Bayes}$

O espaço de Hipóteses

- O espaço de hipóteses é completo
  - Qualquer função pode ser representada por uma árvore de decisão.
- Não reconsidera opções tomadas
  - Mínimos locais
- Escolhas com suporte estatístico
  - Robusto ao ruído
- Preferência por árvores mais pequenas



Data Streams

- Automatic, high-speed, detailed
  - 3 billion telephone calls per day
  - 30 billion emails per day
  - 1 billion SMS
  - Satellite Data
  - IP Network Traffic
  - ….

	Traditional	Stream
Nr. Of Passes	Multiple	Single
Time	Unlimited	Restrict
Memory	Unlimited	Restrict
Result	Accurate	Approximate

Challenges

- Data is collected continuously over time
  - Finances, Economics, Telecommunications, ….
  - Huge volumes of data
- Most of data-mining techniques are memory based
  - All the data must be resident in main-memory
- Our goal
  - Design incremental algorithms that work online
    - Given the actual decision model and a new example modify the actual model to accommodate the example.
  - Today's talk: focus on classification problems
    - Given a *infinite* sequence of pairs of the form $\{\vec{x}_i, y_i\}$
      - Where $y \in \{y_1, y_2, \ldots, y_n\}$
    - Find a function $y = f(x)$
    - That can predict the y value for an unseen $\vec{x}$

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
    - Without (or reduced) revisit old records.
    - 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

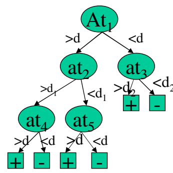
Related Work

- 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
    - Assistant (I. Kononenko), 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

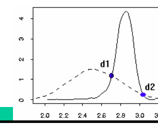
Binary decision trees for data streams



- 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

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

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

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

where

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

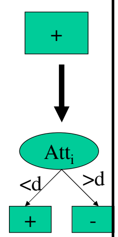
- The attributes are sorted by information gain.
  - $G(X_a)>G(X_b)> ...> G(X_c)$
- When should we transform a leaf into a decision node?
  - When there is a high probability that the selected attribute is the wright one !

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, $G(X_a)>G(X_b)> ...> G(X_c)$
- 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:

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

Functional Nodes

- Each leaf of a tree maintain a naïve Bayes classifier
- When evaluating the splitting criteria
    - After seeing n_{min} examples
  - If there is a tie in the first evaluation
    - Following examples will be classified using the naïve Bayes
    - A contingency table is constructed
      - Naïve Bayes prediction was TRUE or FALSE
    - Next evaluation considers the predictions of naïve Bayes as a pseudo-attribute
    - If this is the best attribute and satisfies the Hoeffding bound
      - It is chosen as test attribute
        - » The outcomes are the prediction of naïve Bayes

		Predicted	
		+	--
Observed	+	n++	n+-
	--	n-+	n--

$$\log(P(Cl_i)) + \sum_j \log(\phi(\bar{x}_{ij}^k, \sigma_{ij}^k))$$

NB predicts +

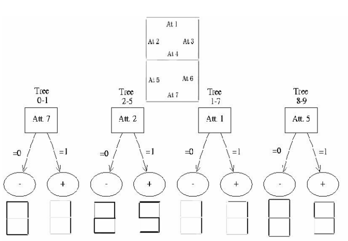
NB predicts -

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.

Example of a Forest

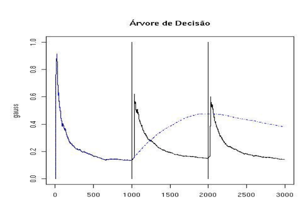
LED Dataset
10 classes
45 decision trees



–Classifying a test example
•Three classes A,B,C
•Three decision Trees
–T1: (A-B),
–T2: (A-C),
–T3: (B-C)
»Suppose the outputs:
»T1 (0.9,0.1),
»T2 (1,0),
»T3 (0.6,0.4)
»The Sum Rule:
»(1.9, 0.7,0.4)
»Final Prediction
»(0.63,0.23,0.13)

Concept Drift

- Goal
  - Online Learning in the context of non-stationary data
- The Basic Idea:
  - When there is a change in the class-distribution of the examples:
    - The actual model does not correspond any more to the actual distribution
      - The error-rate increase



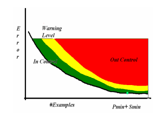
The Method

- At each node of the Tree maintain a naïve-Bayes Classifier
  - Directly derived from the statistics needed by the splitting criteria
  - When an example traverse a node, the naïve-Bayes classifies the example
  - Given a sequence of training examples, the predictions of naïve-Bayes are Bernoulli experiments:
    - T,F,T,F,T,F,T,T,T,F,….
    - With
      - $p_i = (\#F/i)$
      - $S_i = \sqrt{p_i(1-p_i)/i}$
        - » *Where i is the number of trials*

$\forall i$ in the actual context

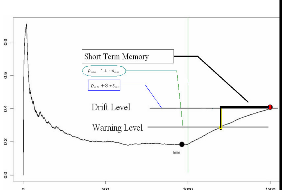
Detect Drift

- The algorithm maintains two registers
  - P_{min} and S_{min} such that $P_{min}+S_{min} = \min(p_i+s_i)$
    - Minimum of the Error rate taking the variance of the estimator into account.
- At example *j*
  - The error of the learning algorithm will be
    - **Out-control** if $p_j+s_j > p_{min}+ \alpha * s_{min}$
    - **In-control** if $p_j+s_j < p_{min}+ \beta * s_{min}$
    - Warning if $p_{min}+ \alpha * s_{min} > p_j+s_j > p_{min}+ \beta * s_{min}$
      - » The constants α and β depend on the confidence level
      - » In our experiments $\beta=2$ and $\alpha = 3$

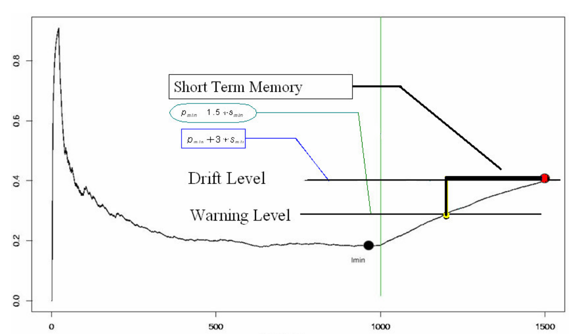


The Algorithm

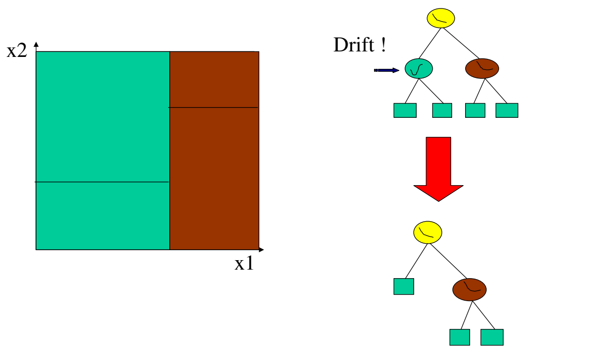
- At example *j* the actual model classifies the example
  - Compute the error and variance: p_j and s_j
  - If the error is
    - In-control the actual model is updated
      - Incorporate the example in the decision model
    - Warning zone:
      - Maintain the actual model
      - First Time:
        - » the lower limit of the window is:
        - » $L_{warning} = j$
    - Out-Control
      - Re-learn a new model using as training set the set of examples $[L_{warning}, j]$



Concept Drift



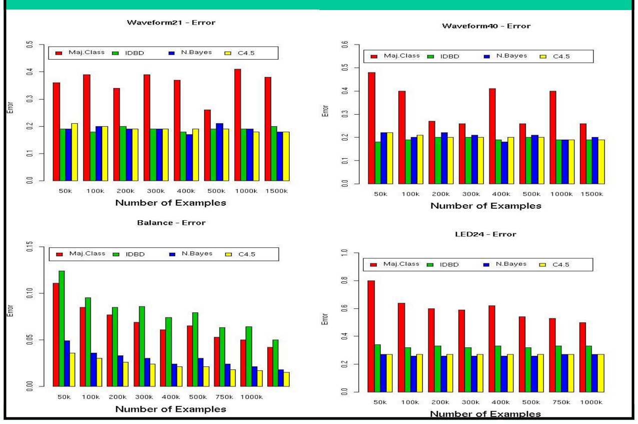
Detecting Drift: pruning nodes



Experimental Evaluation

- The algorithm has been implemented and evaluated.
- Four data streams
  - Electricity Market Dataset
  - Waveform. Two data streams (21 attributes, 40 attributes)
    - Bayes Error 16%
  - LED (24 attributes, 17 irrelevant)
    - Bayes Error: 26%
  - Balance Scale (4 Attributes, 3 Classes)
- Evaluation Criteria: error on an independent test set
- Goals:
  - Comparative study of UFFT versus a standard batch decision tree learner (C4.5)
    - Error Rate
    - Learning Times
    - Tree Size
  - Study the effect of Functional leaves in terms of error rate
  - Sensitivity to
    - Order of examples
    - Noise

Learning Curves: Error Rate vs. Nr of Examples

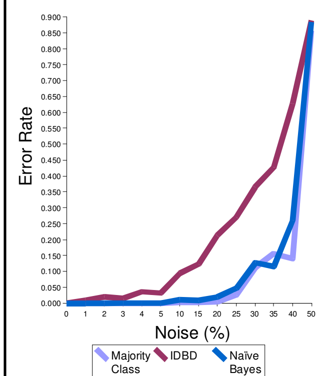


Training Time vs. Nr of Examples

Learning Curves: Error Rate versus Nr. of Examples

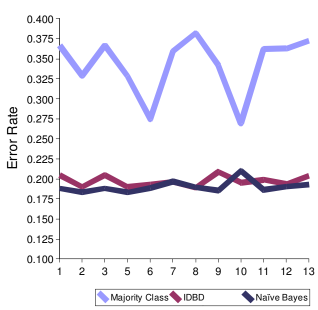
- Using the majority class at leaves:
  - The error rate decreases when training set size increases
- Using Functional leaves:
  - We observe strong improvements of the error rate.
  - The performance of any Functional model is quite similar to a standard batch tree learner.
  - The error rate is almost constant
    - **Anytime** classifier

Sensitivity to Noise



- Design of Experiments:
  - Dataset: LED24
  - Test Dataset without noise
    - 100.000 examples
  - Training set
    - 200.000 examples
    - Noise in Training set varying from 0% to 50%
- The performance of UFFT is dependent of the classification strategy at leaves:
  - MC and NB similar behaviors
    - Less effected
    - IDBD very sensible

Sensitivity to the Order of Examples



- Design of experiments
  - Dataset: Waveform 40
    - Fixed training set: 300,000
    - Train UFFT with different permutations of the training set
      - Changes in the order of the examples
    - Fixed Test set: 250,000
- The performance of UFFT has low dependence from the order of the examples
  - Naïve Bayes & IDBD with very low sensitivity to the order of the sample
  - Majority class is the most effected.

Drift Evaluation

Artificial Data:

1	800	1600	2400

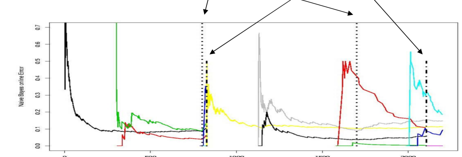
Concept 1	Concept 2	Concept 3
Att1 > 0.5 **Att1 > Att2**	**Att1 < 0.5** **Att1 < Att2**	**Att1 < 0.4** **Att1 < 2.5 * Att2**

Evaluation:

(Independent Test set drawn from concept3):

Drift Detection:	3%
Without Drift Detection:	16%

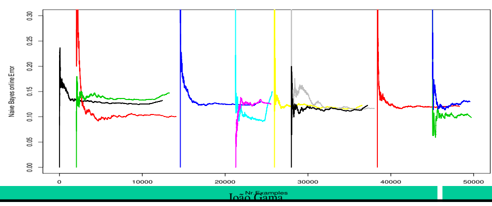
Drift Occurs   Drift Detect



SEA Concepts

Streaming Ensemble Algorithm for large scale classification, N. Street, Y.Kim KDD01

	UFFT	UFFT-ND	CVFDT	VFDT	VFDTc
Error	12.99	15.89	14.72	16.06	14.40
t.test	-	0	0.002	0	0.0001



The Electricity Market Dataset

- The data was collected from the Australian New South Wales Electricity Market
  - The electricity price is not fixed
    - The price is set every 5 minutes
    - It is affected by demand and supply of the market
  - The dataset covers the period from 7 May 1996 till 5 December 1998
    - Contains 45312 examples
    - Attributes
      - Day of Week
      - NSW electricity demand
      - Victorian electricity demand
      - Scheduled electricity transfer
      - …
      - Class Label:
        - » Change (UP, DOWN) of the price related to a moving average of the last 24 hours.

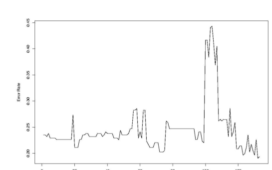
Experiments

- Two sets of experiments:
  - Predicting last week
  - Predicting last day
- Error-rates using the decision tree available in R (CART like):

Test Set	All Data	Last Year
Last Day	18.7%	12.5%
Last Week	23.5%	22.4%

Generalization Bound

- A Lower Bound for the generalization error:
  - Exhaustive search of the best training set
    - *looking to the error in the test set*
  - Training set:
    - Last week: 3548 examples
      - Test Error: 19%
    - Last day: 3836 examples
      - Test Error: 10.4%



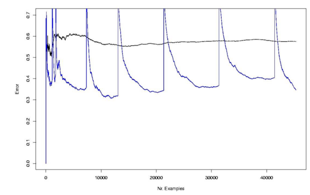
Results using Drift Detection

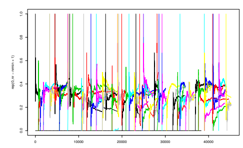
Test Set	Lower Bound	All Data	Last Year	Drift Detection
Last Day	10.4%	18.7%	12.5%	10.4%
Last Week	19.0%	23.5%	22.4%	19.9%

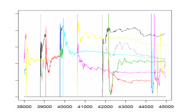
Online error

Trace of the online error of a decision tree:

- Using drift detection
- Without using drift detection







Conclusions

- UFFT: Incremental, online forest of trees for data-streams
  - Processes each example in constant time and memory
  - Single scan over the data
  - Functional Leaves
    - Anytime Classifier
- The experimental section suggests:
  - Performance similar to a batch decision tree learner when using Functional leaves.
  - No need for pruning.
    - Decisions with statistical support.
  - Resilience to the order of examples, noise
  - Robust to detect concept drift
- Future Work
  - Multivariate decision nodes

Thanks for your attention!

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