Classification

Albert Bifet

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

April 2012

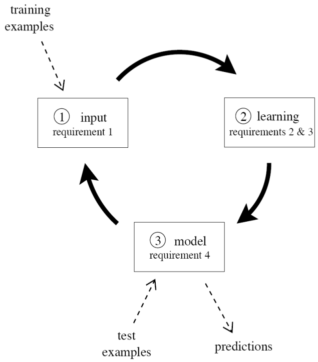
Outline

Data Streams

Big Data & Real Time

Data stream classification cycle

1. Process an example at a time, and inspect it only once (at most)
2. Use a limited amount of memory
3. Work in a limited amount of time
4. Be ready to predict at any point



Classification

Definition

Given n_C different classes, a classifier algorithm builds a model that predicts for every unlabelled instance I the class C to which it belongs with accuracy.

Example

A spam filter

Example

Twitter Sentiment analysis: analyze tweets with positive or negative feelings

Bayes Classifiers

Naïve Bayes

- Based on Bayes Theorem:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

$$posterior = \frac{prior \times likelikood}{evidence}$$

- Estimates the probability of observing attribute a and the prior probability $P(c)$
- Probability of class c given an instance d:

$$P(c|d) = \frac{P(c)\prod_{a \in d} P(a|c)}{P(d)}$$

Bayes Classifiers

Multinomial Naïve Bayes

- Considers a document as a bag-of-words.
- Estimates the probability of observing word w and the prior probability $P(c)$
- Probability of class c given a test document d:

$$P(c|d) = \frac{P(c)\prod_{w \in d} P(w|c)^{n_{wd}}}{P(d)}$$

Classification

Example

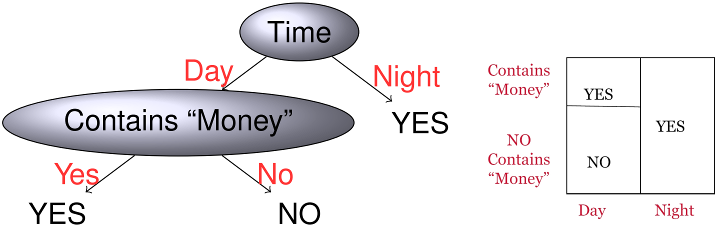
Data set for sentiment analysis

Id	Text	Sentiment
T1	glad happy glad	+
T2	glad glad joyful	+
T3	glad pleasant	+
T4	miserable sad glad	-

Assume we have to classify the following new instance:

Id	Text	Sentiment
T5	glad sad miserable pleasant sad	?

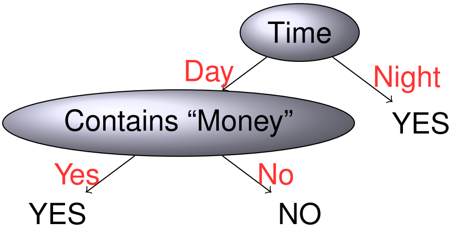
Decision Tree

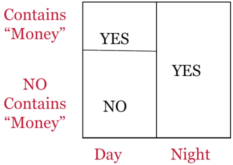


Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to an attribute value
- Each leaf node assigns a classification

Decision Tree





Main loop:

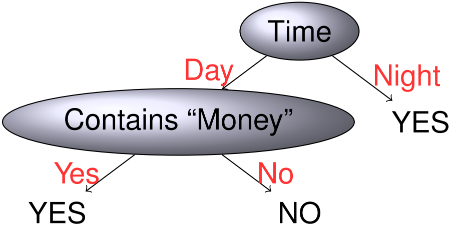
- $A \leftarrow$ the “best” decision attribute for next *node*
- Assign A as decision attribute for *node*
- For each value of A, create new descendant of *node*
- Sort training examples to leaf nodes
- If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Hoeffding Trees

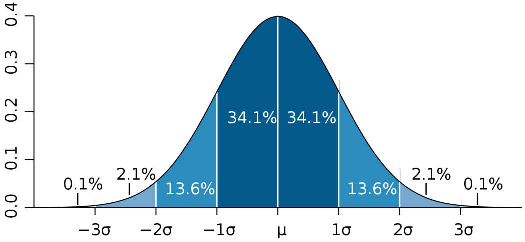
Hoeffding Tree : VFDT

Pedro Domingos and Geoff Hulten.
Mining high-speed data streams. 2000

- With high probability, constructs an identical model that a traditional (greedy) method would learn
- With theoretical guarantees on the error rate



Hoeffding Bound Inequality



Probability of deviation of its expected value.

Hoeffding Bound Inequality

Let $X = \sum_i X_i$ where $X_1, \ldots, X_n$ are independent and indentically distributed in $[0, 1]$. Then

1. **Chernoff** For each $\epsilon < 1$

$$\Pr[X > (1+\epsilon)E[X]] \leq \exp\left(-\frac{\epsilon^2}{3}E[X]\right)$$

2. **Hoeffding** For each $t > 0$

$$\Pr[X > E[X] + t] \leq \exp\left(-2t^2/n\right)$$

3. **Bernstein** Let $\sigma^2 = \sum_i \sigma_i^2$ the variance of X. If $X_i - E[X_i] \leq b$ for each $i \in [n]$ then for each $t > 0$

$$\Pr[X > E[X] + t] \leq \exp\left(-\frac{t^2}{2\sigma^2 + \frac{2}{3}bt}\right)$$

Hoeffding Tree or VFDT

HT($Stream, \delta$)

$\triangleright$ Let HT be a tree with a single leaf(root)
$\triangleright$ Init counts n_{ijk} at root
for each example (x, y) in Stream
    do HTGROW($(x, y), HT, \delta$)

Hoeffding Tree or VFDT

HT($Stream, \delta$)

▷ Let HT be a tree with a single leaf(root)
▷ Init counts n_{ijk} at root
for each example (x, y) in Stream
    do HTGROW($(x, y), HT, \delta$)

HTGROW($(x, y), HT, \delta$)

▷ Sort (x, y) to leaf l using HT
▷ Update counts n_{ijk} at leaf l
if examples seen so far at l are not all of the same class
    then ▷ Compute G for each attribute
        if $G(\text{Best Attr.}) - G(\text{2nd best}) > \sqrt{\frac{R^2 \ln 1/\delta}{2n}}$
            then ▷ Split leaf on best attribute
                for each branch
                    do ▷ Start new leaf and initiliatize counts

Hoeffding Trees

HT features

- With high probability, constructs an identical model that a traditional (greedy) method would learn
- Ties: when two attributes have similar G, split if

$$G(\text{Best Attr.}) - G(\text{2nd best}) < \sqrt{\frac{R^2 \ln 1/\delta}{2n}} < \tau$$

- Compute G every n_{min} instances
- Memory: deactivate least promising nodes with lower $p_l \times e_l$
  - p_l is the probability to reach leaf l
  - e_l is the error in the node

Hoeffding Naive Bayes Tree

Hoeffding Tree

Majority Class learner at leaves

Hoeffding Naive Bayes Tree

G. Holmes, R. Kirkby, and B. Pfahringer.
Stress-testing Hoeffding trees, 2005.

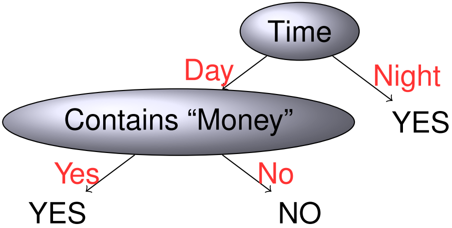
- monitors accuracy of a Majority Class learner
- monitors accuracy of a Naive Bayes learner
- predicts using the most accurate method

Decision Trees: CVFDT

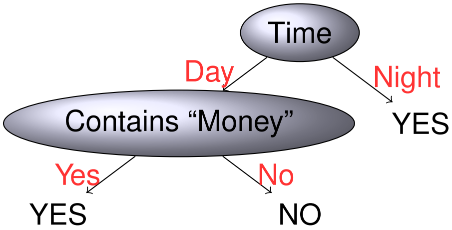
Concept-adapting Very Fast Decision Trees: CVFDT

G. Hulten, L. Spencer, and P. Domingos.
Mining time-changing data streams. 2001

- It keeps its model consistent with a sliding window of examples
- Construct “alternative branches” as preparation for changes
- If the alternative branch becomes more accurate, switch of tree branches occurs



Decision Trees: CVFDT

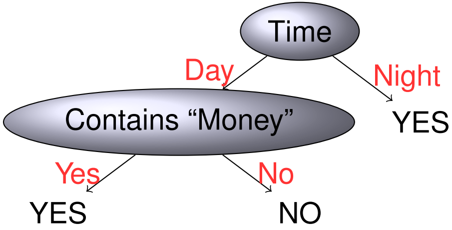


No theoretical guarantees on the error rate of CVFDT

CVFDT parameters :

1. W: is the example window size.
2. T_0: number of examples used to check at each node if the splitting attribute is still the best.
3. T_1: number of examples used to build the alternate tree.
4. T_2: number of examples used to test the accuracy of the alternate tree.

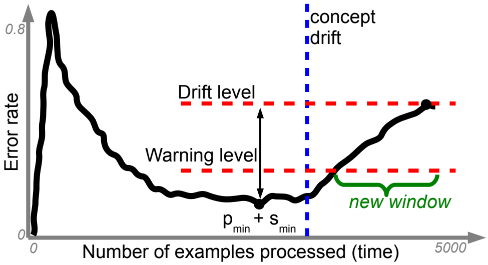
Concept Drift: VFDTc (Gama et al. 2003,2006)



VFDTc improvements over HT:

1. Naive Bayes at leaves
2. Numeric attribute handling using BINTREE
3. Concept Drift Handling: Statistical Drift Detection Method

Concept Drift



Statistical Drift Detection Method
(Gama et al. 2004)

Decision Trees: Hoeffding Adaptive Tree

Hoeffding Adaptive Tree:

- replace frequency statistics counters by estimators
  - don't need a window to store examples, due to the fact that we maintain the statistics data needed with estimators
- change the way of checking the substitution of alternate subtrees, using a change detector with theoretical guarantees (`ADWIN`)

Advantages over CVFDT:

1. Theoretical guarantees
2. No Parameters

Numeric Handling Methods

VFDT (VFML – Hulten & Domingos, 2003)

- Summarize the numeric distribution with a histogram made up of a maximum number of bins N (default 1000)
- Bin boundaries determined by first N unique values seen in the stream.
- Issues: method sensitive to data order and choosing a good N for a particular problem

Exhaustive Binary Tree (BINTREE – Gama et al, 2003)

- Closest implementation of a batch method
- Incrementally update a binary tree as data is observed
- Issues: high memory cost, high cost of split search, data order

Numeric Handling Methods

Quantile Summaries (GK – Greenwald and Khanna, 2001)

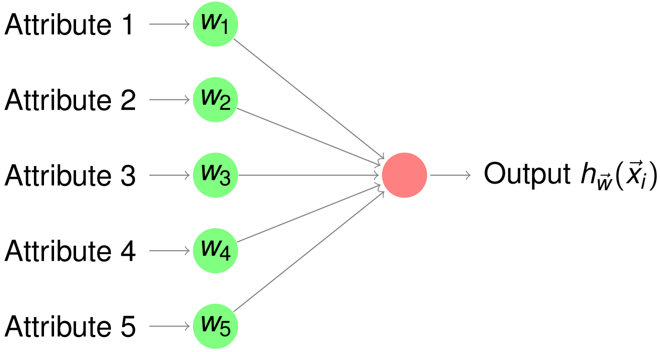
- Motivation comes from VLDB
- Maintain sample of values (quantiles) plus range of possible ranks that the samples can take (tuples)
- Extremely space efficient
- Issues: use max number of tuples per summary

Numeric Handling Methods

Gaussian Approximation (GAUSS)

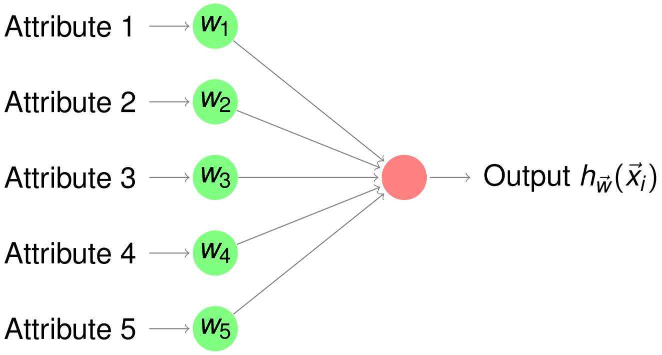
- Assume values conform to Normal Distribution
- Maintain five numbers (eg mean, variance, weight, max, min)
- Note: not sensitive to data order
- Incrementally updateable
- Using the max, min information per class – split the range into N equal parts
- For each part use the 5 numbers per class to compute the approx class distribution
- Use the above to compute the IG of that split

Perceptron



- Data stream: $\langle \vec{x}_i, y_i \rangle$
- Classical perceptron: $h_{\vec{w}}(\vec{x}_i) = \mathrm{sgn}(\vec{w}^T \vec{x}_i)$,
- Minimize Mean-square error: $J(\vec{w}) = \frac{1}{2} \sum (y_i - h_{\vec{w}}(\vec{x}_i))^2$

Perceptron



- We use sigmoid function $h_{\vec{w}} = \sigma(\vec{w}^T \vec{x})$ where

$$\sigma(x) = 1/(1 + e^{-x})$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Perceptron

- Minimize Mean-square error: $J(\vec{w}) = \frac{1}{2}\sum(y_i - h_{\vec{w}}(\vec{x}_i))^2$
- Stochastic Gradient Descent: $\vec{w} = \vec{w} - \eta\nabla J\vec{x}_i$
- Gradient of the error function:

$$\nabla J = -\sum_i (y_i - h_{\vec{w}}(\vec{x}_i))\nabla h_{\vec{w}}(\vec{x}_i)$$

$$\nabla h_{\vec{w}}(\vec{x}_i) = h_{\vec{w}}(\vec{x}_i)(1 - h_{\vec{w}}(\vec{x}_i))$$

- Weight update rule

$$\vec{w} = \vec{w} + \eta\sum_i (y_i - h_{\vec{w}}(\vec{x}_i))h_{\vec{w}}(\vec{x}_i)(1 - h_{\vec{w}}(\vec{x}_i))\vec{x}_i$$

Perceptron

PERCEPTRON LEARNING(*Stream*, η)

for each class
    do PERCEPTRON LEARNING(*Stream*, *class*, η)

PERCEPTRON LEARNING(*Stream*, *class*, η)

$\triangleright$ Let w_0 and $\vec{w}$ be randomly initialized
for each example $(\vec{x}, y)$ in Stream
    do if $class = y$
        then $\delta = (1 - h_{\vec{w}}(\vec{x})) \cdot h_{\vec{w}}(\vec{x}) \cdot (1 - h_{\vec{w}}(\vec{x}))$
        else $\delta = (0 - h_{\vec{w}}(\vec{x})) \cdot h_{\vec{w}}(\vec{x}) \cdot (1 - h_{\vec{w}}(\vec{x}))$
      $\vec{w} = \vec{w} + \eta \cdot \delta \cdot \vec{x}$

PERCEPTRON PREDICTION($\vec{x}$)

return $\arg\max_{class} h_{\vec{w}_{class}}(\vec{x})$

Multi-label classification


- Binary Classification: e.g. is this a beach? $\in \{\texttt{No, Yes}\}$
- Multi-class Classification: e.g. what is this?
  $\in \{\texttt{Beach, Forest, City, People}\}$
- Multi-label Classification: e.g. which of these?
  $\subseteq \{\texttt{Beach, Forest, City, People}\}$

Methods for Multi-label Classification

Problem Transformation: Using off-the-shelf binary / multi-class classifiers for multi-label learning.

- **Binary Relevance method (BR)**
  - One binary classifier for each label:
    - simple; flexible; fast but does not explicitly model label dependencies
- **Label Powerset method (LP)**
  - One multi-class classifier; one class for each labelset

Data Streams Multi-label Classification

- **Adaptive Ensembles of Classifier Chains (ECC)**
  - Hoeffding trees as base-classifiers
  - reset classifiers based on current performance / concept drift
- **Multi-label Hoeffding Tree**
  - Label Powerset method (LP) at the leaves an ensemble strategy to deal with concept drift
  - $\mathrm{entropy}_{SL}(S) = -\sum_{i=1}^{N} p(i) \log(p(i))$

$$\mathrm{entropy}_{ML}(S) = \mathrm{entropy}_{SL}(S) - \sum_{i=1}^{N} (1 - p(i)) \log(1 - p(i))$$

Active Learning

ACTIVE LEARNING FRAMEWORK

Input: labeling budget B and strategy parameters

```
for each X_t - incoming instance,
    do if ACTIVE LEARNING STRATEGY(X_t, B, ...) = true
        then request the true label y_t of instance X_t
            train classifier L with (X_t, y_t)
            if L_n exists then train classifier L_n with (X_t, y_t)
            if change warning is signaled
               then start a new classifier L_n
            if change is detected
               then replace classifier L with L_n
```

Active Learning

	Controlling Budget	Instance space Coverage
Random	present	full
Fixed uncertainty	no	fragment
Variable uncertainty	handled	fragment
Randomized uncertainty	handled	full

Table : Summary of strategies.