Lecture notes

Knowledge Discovery in Databases II

Winter Semester 2012/2013

Lecture 5: Stream classification

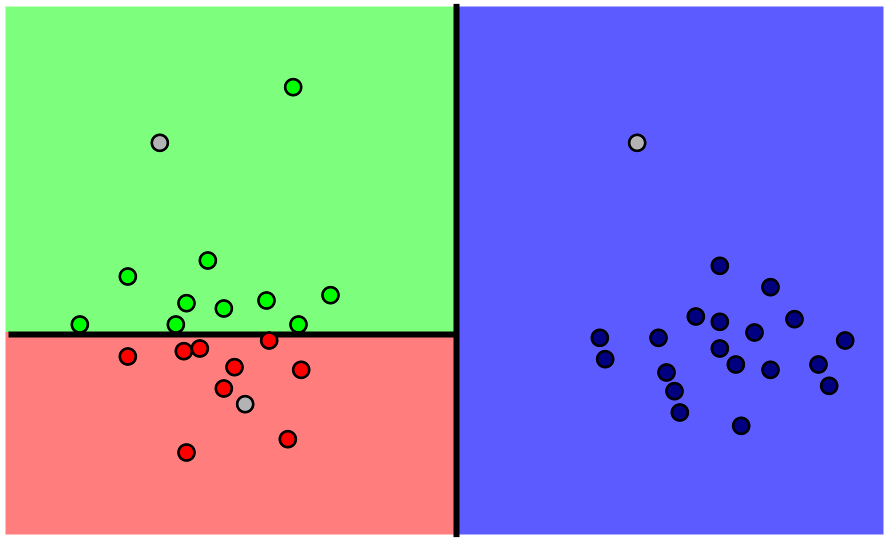
Lectures: PD Dr Matthias Schubert, Dr. Eirini Ntoutsi
Tutorials: Erich Schubert

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

Outline

- Introduction
- Data stream classification
- Data stream classifiers
- Data stream classifiers evaluation
- Things you should know
- Homework/tutorial

Classification



- ● Screw
- ● Nails
- ● Paper clips

} Training data

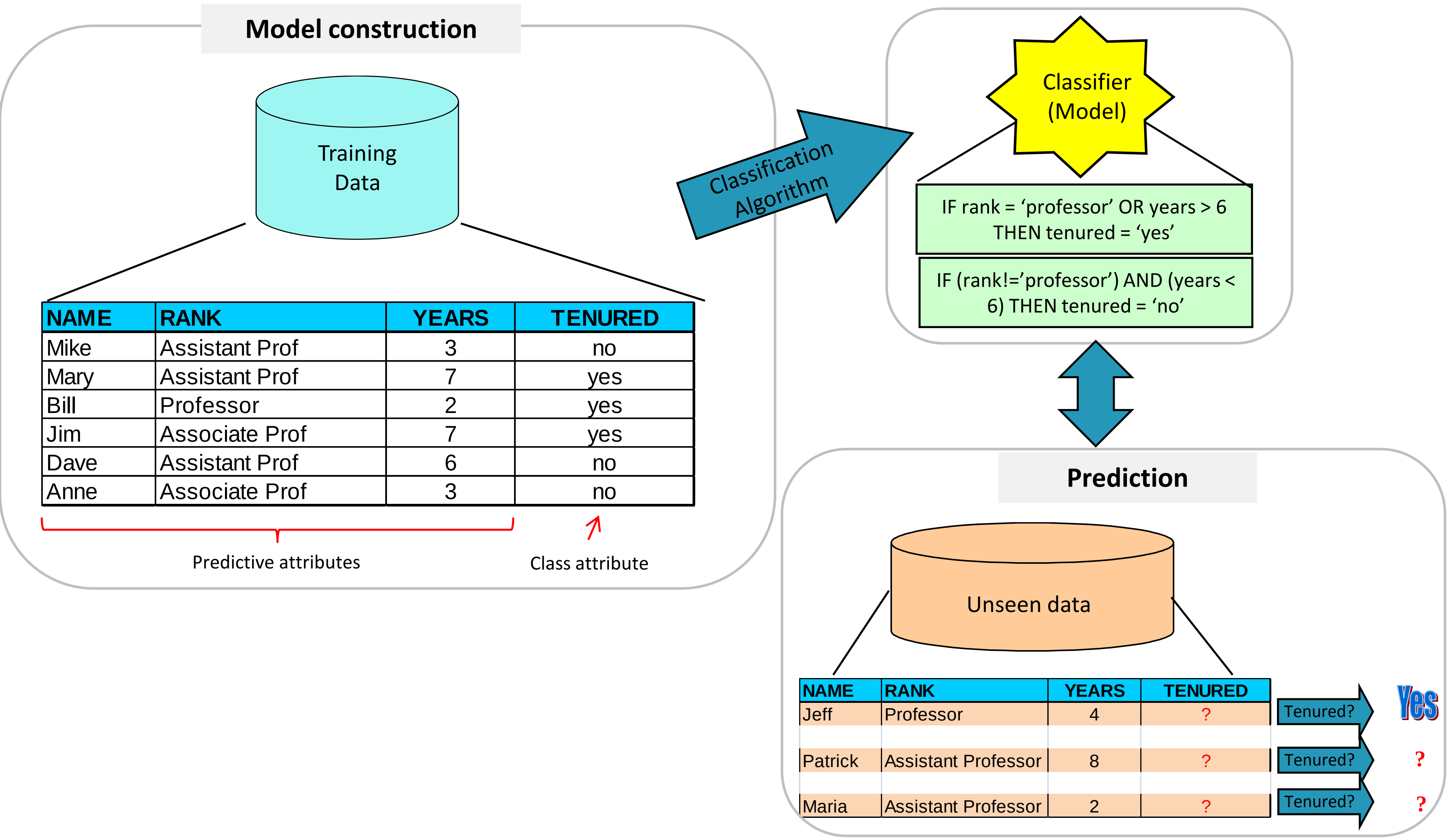
- ● New object

<u>Task:</u>

Learn from the already classified training data, the rules to classify new objects based on their characteristics.

The result attribute (class variable) is nominal (categorical)

The (batch) classification process



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor' OR years > 6 THEN tenured = 'yes'

IF (rank!='professor') AND (years < 6) THEN tenured = 'no'

NAME	RANK	YEARS	TENURED
Jeff	Professor	4	?
Patrick	Assistant Professor	8	?
Maria	Assistant Professor	2	?

Outline

- Introduction
- Data stream classification
- Data stream classifiers
- Data stream classifiers evaluation
- Things you should know
- Homework/tutorial

Classification over dynamic data I

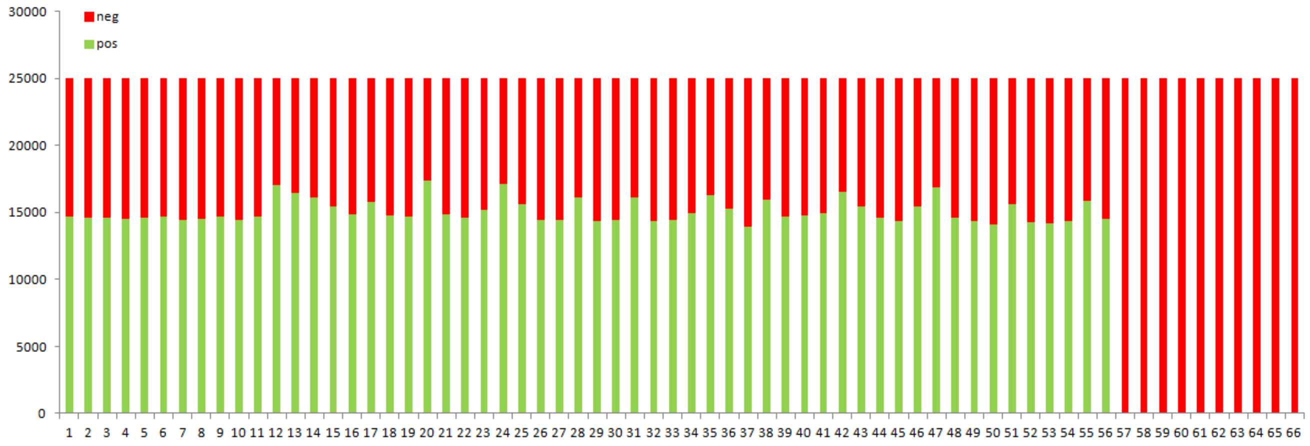
- So far, classification as a batch/ static task
  - The whole training set is given as input to the algorithm for the generation of the classification model
  - When the performance of the model drops, a new model is generated from scratch over a new training set
- But, in a dynamic environment data change continuously
  - Batch model re-generation is not appropriate/sufficient anymore
- Need for new classification algorithms that
  - update existing models by incorporating new data
  - deal with non-stationary data generation processes (concept drift)
  - subject to:
    - o Resource constraints (processing time, memory)
    - o Single scan of the data (one look, no random access)

DATABASE SYSTEMS GROUP

Classification over dynamic data II

LMU

- Non-stationary data (evolving distribution)



Evolving data distribution (from the BA of Alina's Sinelnikova, "Sentiment analysis in the Twitter stream", LMU 2012.)

- Concept drift

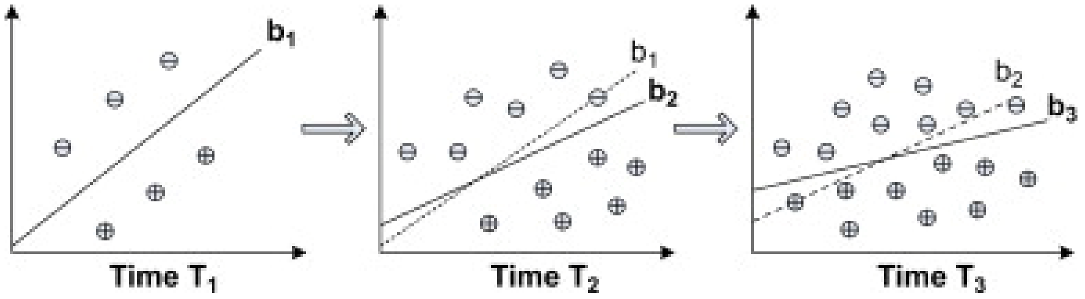


Fig. 1. An illustration of concept drifting in data streams. In the three consecutive time stamps T_1, T_2 and T_3, the classification boundary gradually drifts from b_1 to b_2 and finally to b_3.
(from: A framework for application-driven classification of data streams, Zhang et al, Journal Neurocomputing 2012)

Data stream classifiers

- The batch classification problem:
  - Given a finite training set D={(x,y)} , where y={y_1, y_2, …, y_k}, |D|=n, find a function y=f(x) that can predict the y value for an unseen instance x
- The data stream classification problem:
  - Given an infinite sequence of pairs of the form (x,y) where y={y_1, y_2, …, y_k}, find a function y=f(x) that can predict the y value for an unseen instance x
- Example applications:
  - Fraud detection in credit card transactions
  - Churn prediction in a telecommunication company
  - Sentiment classification in the Twitter stream
  - Topic classification in a news aggregation site, e.g. Google news
  - …

Outline

- Introduction
- Data stream classification
- Data stream classifiers
- Data stream classifiers evaluation
- Things you should know
- Homework/tutorial

Data stream classifiers

- Decision trees
- Ensemble methods
- Naïve Bayes classifiers
- Stochastic Gradient Descent (SGD)
- Neural Nets
- …

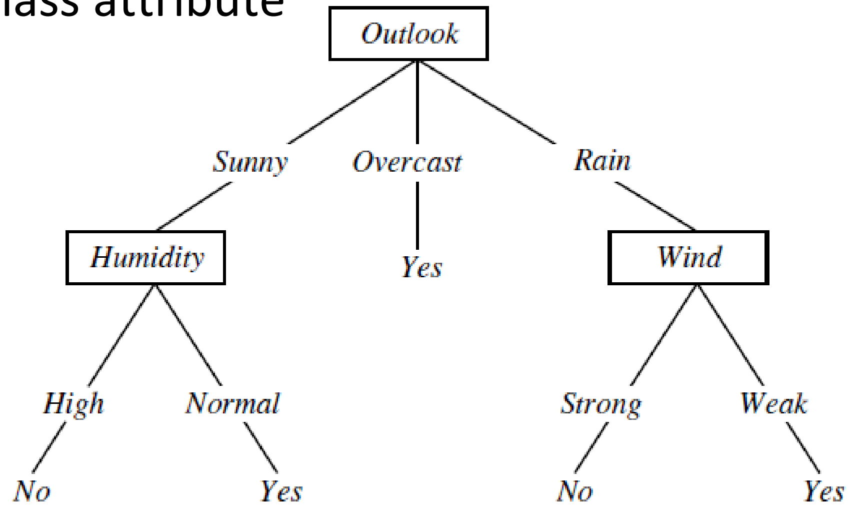
(Batch) Decision Trees (DTs)

- Training set: D = {(x,y)}
  - predictive attributes: $x=<x_1, x_2, \ldots, x_d>$
  - class attribute: $y=\{y_1, y_2, \ldots, y_k\}$
- Goal: find y=f(x)
- Decision tree model
  - nodes contain tests on the predictive attributes
  - leaves contain predictions on the class attribute

Training set

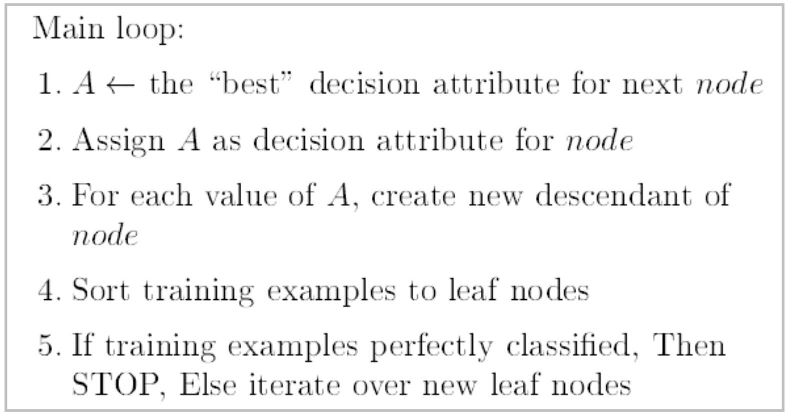
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$\Rightarrow$



(Batch) DT induction algorithm

- Basic algorithm (ID3, Quinlan 1986)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root node

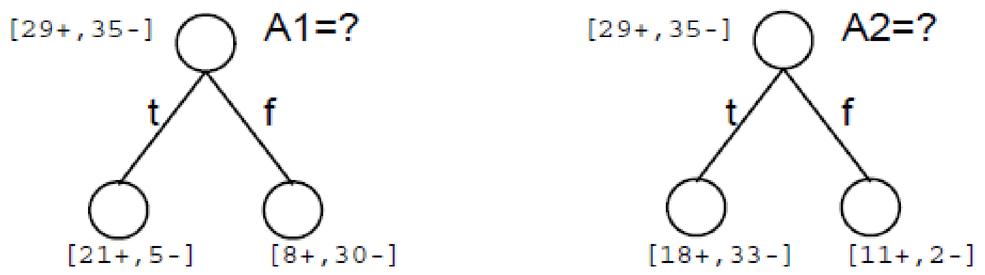


```
Main loop:
1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign $A$ as decision attribute for *node*
3. For each value of $A$, create new descendant of
   *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then
   STOP, Else iterate over new leaf nodes
```

  - But, which attribute is the best?

Attribute selection measures:
- Information gain
- Gain ratio
- Gini index

(check Lecture 4, KDD I)



Goal: select the most useful attribute for classifying examples.
• useful → the resulting partitioning is as pure as possible
• pure partition: all its instances belong to the same class.

(Batch) DTs: Information gain

- Used in ID3
- It uses entropy, a measure of pureness of the data
- The information gain Gain(S,A) of an attribute A relative to a collection of examples S measures the gain reduction in S due to splitting on A:

$$Gain(S,A) = \underbrace{\boxed{Entropy(S)}}_{\text{Before splitting}} - \underbrace{\boxed{\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}}_{\text{After splitting on A}}$$

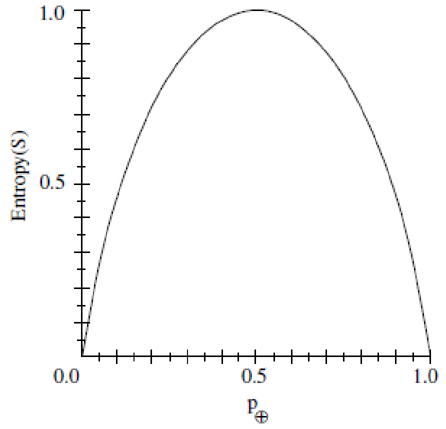
- Gain measures the expected reduction in entropy due to splitting on A

- The attribute with the higher entropy reduction is chosen

(Batch) DTs: Entropy

- Let S be a collection of positive and negative examples for a binary classification problem, C={+, -}.
- p_+: the percentage of positive examples in S
- p_-: the percentage of negative examples in S
- Entropy measures the impurity of S:

$$Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$



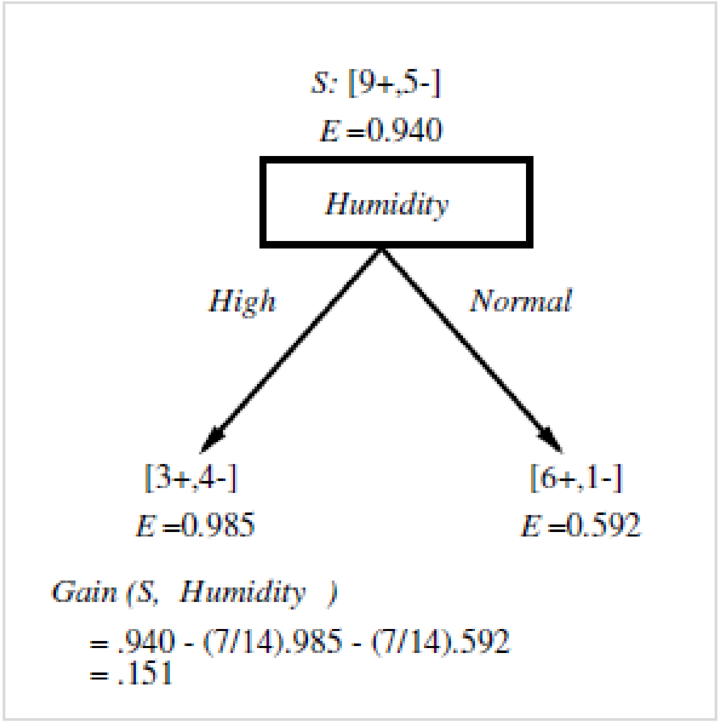
in the general case
(k-classification problem)

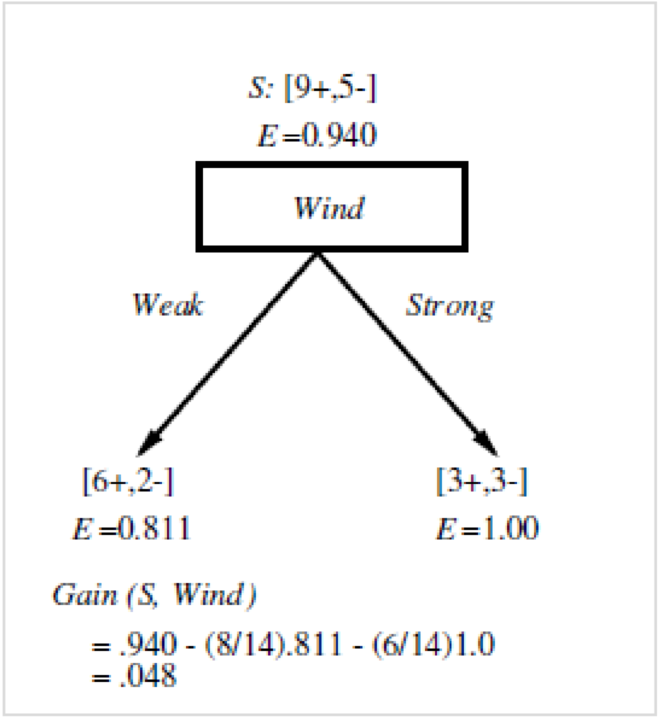
$$Entropy(S) = \sum_{i=1}^{k} -p_i \log_2(p_i)$$

- Examples :
  - Let S: [9+,5-] $Entropy(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$
  - Let S: [7+,7-] $Entropy(S) = -\frac{7}{14}\log_2(\frac{7}{14}) - \frac{7}{14}\log_2(\frac{7}{14}) = 1$
  - Let S: [14+,0-] $Entropy(S) = -\frac{14}{14}\log_2(\frac{14}{14}) - \frac{0}{14}\log_2(\frac{0}{14}) = 0$
- Entropy = 0, when all members belong to the same class
- Entropy = 1, when there is an equal number of positive and negative examples

(Batch) DTs: Information gain example

- Which attribute to choose next?





Hoeffding trees

- Idea: In order to pick the best split attribute for a node, it may be sufficient to consider only a small subset of the training examples that pass through that node.
  - No need to look at the whole dataset (which is infinite in case of streams)
  - E.g., use the first few examples to choose the split at the root
- Problem: How many instances are necessary?
  - Hoeffding bound!
- Hoeffding tree variations
  - Very Fast DTs (VFDT), Domingos and Hulten, KDD 2000.
  - VFDTc, Gamma et al, KDD 2003
  - …

Hoeffding bound I

- Consider a real-valued random variable r whose range is R
  - e.g., for a probability the range is one,
  - for an information gain the range is $\log_2(c)$, where c is the number of classes
- Suppose we have n independent observations of r and we compute its mean $\bar{r}$
- The Hoeffding bound states that with probability 1-δ the true mean of the variable μ_r will not differ by more than ε from the estimated mean after n independent observations, i.e., $P(\mu_r \geq \bar{r} - \varepsilon) = 1-\delta$, where:

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

- This bound holds true regardless of the distribution generating the values, and depends only on the range of values, number of observations and desired confidence.
  - A disadvantage of being so general is that it is more conservative than a distribution-dependent bound

Hoeffding bound II

- Let G() be the heuristic measure for choosing the split attribute at a node
- After seeing n instances at this node, let
  - X_a : be the attribute with the highest observed G()
  - X_b : be the attribute with the second-highest observed G()
- $\overline{\Delta G} = \overline{G}(X_a) - \overline{G}(X_b) \geq 0$ the difference between the 2 best attributes
- $\overline{\Delta G}$ is the random variable being estimated by the Hoeffding bound
- Given a desired δ, the Hoeffding bound guarantees that with probability 1-δ , X_a is the correct choice for this node, if n instances have been seen so far in this node and $\overline{\Delta G} > \varepsilon$.
  - In this case, the sample size is enough to decide on X_a
- Otherwise, i.e., if $\overline{\Delta G} < \varepsilon$, the sample size is not enough for a stable decision.
  - With R and δ fixed, the only variable left to change ε is n
    - We need to extend the sample by seeing more instances, until ε becomes smaller than $\overline{\Delta G}$

Hoeffding tree induction algorithm

```
Input: $\delta$ desired probability level.
Output: $\mathcal{T}$ A decision Tree
Init: $\mathcal{T} \leftarrow$ Empty Leaf (Root)
While (TRUE)
    Read next Example
    Propagate Example through the Tree from the Root till a leaf
    Update Sufficient Statistics at leaf
    If $leaf(\#examples) > N_{min}$
        Evaluate the merit of each attribute
        Let $A_1$ the best attribute and $A_2$ the second best
        Let $\epsilon = \sqrt{R^2 ln(1/\delta)/(2n)}$
        If $G(A_1) - G(A_2) > \epsilon$
            Install a splitting test based on $A_1$
            Expand the tree with two descendant leaves
```

Those needed by the heuristic evaluation function G()

The evaluation of G() after each instance is very expensive.
$\rightarrow$ Evaluate G() only after N_{min} instances have been observed since the last evaluation.

Replacing a leaf node by a decision node

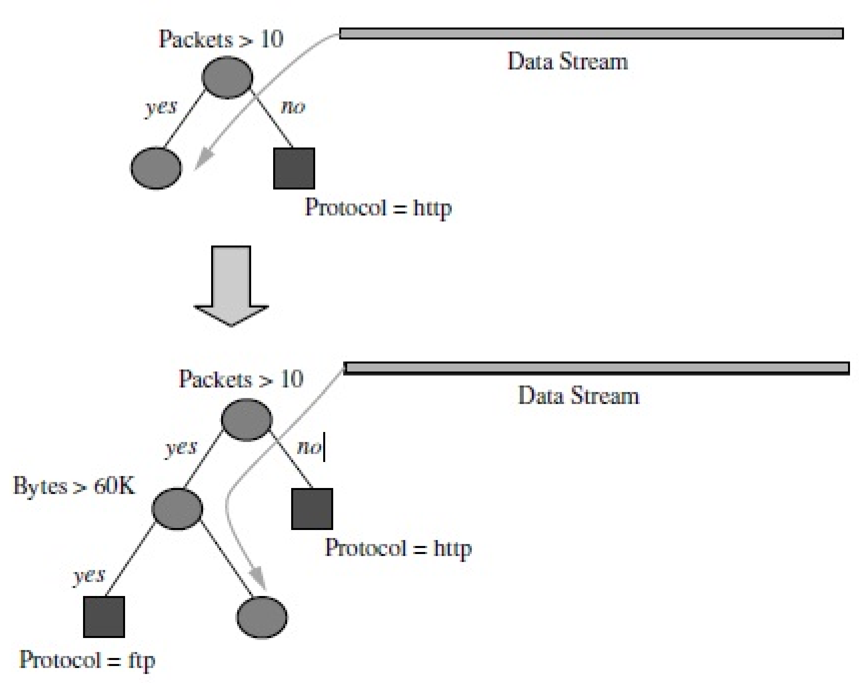


Figure The nodes of the Hoeffding tree are created incrementally as more samples stream in.

Source: http://ptucse.loremate.com/dw/node/5

Hoeffding tree

- Uses limited resources in terms of memory
  - Only storing the necessary statistics at each leaf to decide on split
  - Independent on # instances seen
- It is incremental
- Processes instances only once
- Answers or models are available at any time
- Makes stable decisions with statistical support
  - in contrast to e.g. C4.5 where the number of examples decreases as the tree grows, here all nodes receive the number of examples needed for making a stable decision
- But, it cannot handle concept drift, because once a node is created, it can never change.

Adaptive Size Hoeffding Tree (ASHT) (Bifet et al, KDD 2009)

- Adaptive size Hoeffding tree (ASHT)
  - The tree has a maximum size (# of splitting nodes)
  - After one node splits, if the number of split nodes of the ASHT tree is higher than the maximum value, then it deletes some nodes to reduce its size

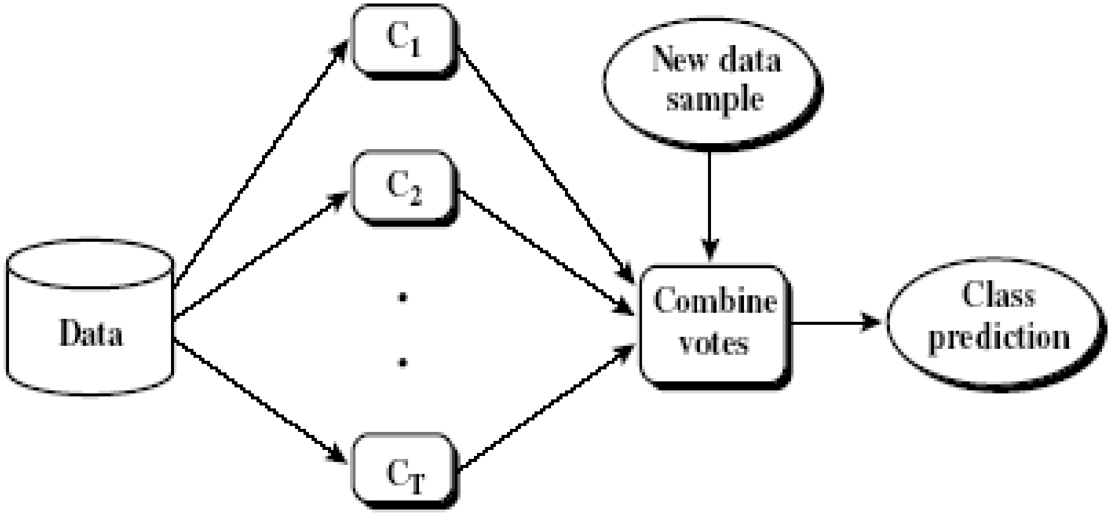
- When the tree size exceeds the maximum size value, there are two different delete options:
  - delete the oldest node, the root, and all of its children except the one where the split has been made. After that, the root of the child not deleted becomes the new root.
  - delete all the nodes of the tree, that is, reset the tree to the empty tree
- It can adapt to data changes

Data stream classifiers

- Decision trees
- Ensemble methods
- Naïve Bayes classifiers
- Stochastic Gradient Descent (SGD)
- Neural Nets
- …

Ensemble of classifiers

- Idea:
  - Instead of a single model, use a combination of models to increase accuracy
  - Combine a series of T learned models, M_1, M_2, …, M_T, with the aim of creating an improved model M*
  - To predict the class of previously unseen records, aggregate the predictions of the ensemble



How to generate ensembles of classifiers

- By manipulating the training set
  - Multiple training sets are created by resampling the original training data
  - A classifier is built from each training set using some learning algorithm
  - e.g., bagging, boosting
- By manipulating the input features
  - A subset of features is chosen to form each training set (randomly or by domain experts)
  - A base classifier is built from each training set using some learning algorithm
  - e.g., random forests
- By manipulating the class labels, by manipulating the learning algorithm,… (see more on Lecture 6, KDD I)

Bagging/ Bootstrap aggregation (Breiman, 1996)

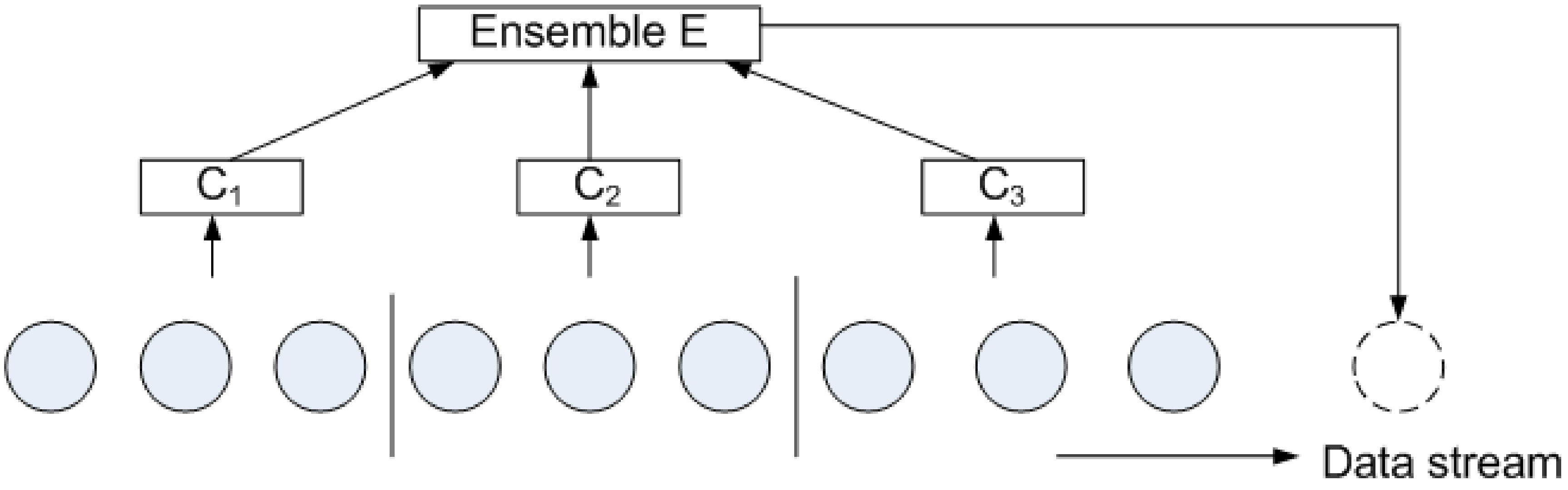
- Analogy: Diagnosis based on multiple doctors' majority vote
- Training: Given a training set D of d tuples
  - In each iteration i: i=1, … , T
    - Randomly sample with replacement from D a training set D_i of *d* tuples (i.e., boostrap)
      - On avg, the bootstrap sample contains approximately 63% of the original D
    - Train a chosen "base model" M_i (e.g. neural network, decision tree) on the sample D_i
- Testing
  - For each test example
    - Get the predicted class from each trained base model M_1, M_2, … M_T
    - Final prediction by majority voting

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
    - Records that are wrongly classified will have their weights increased
    - Records that are classified correctly will have their weights decreased
- Adaptive boosting; each classifier is dependent on the previous one and focuses on the previous one's errors
- Adaboost (Freund and Schapire, 1995)

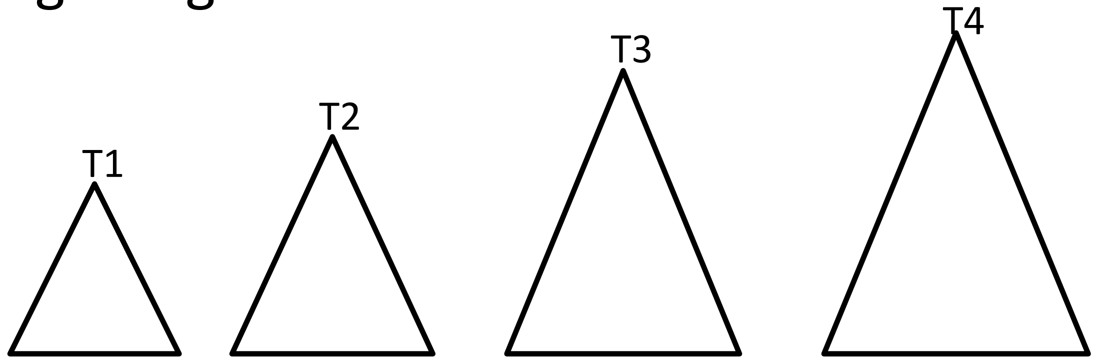
Ensemble Models in streams

- Idea: Use a **divide-and-conquer** manner to build models from continuous data streams



Ensemble of Adaptive Size Hoeffding Trees

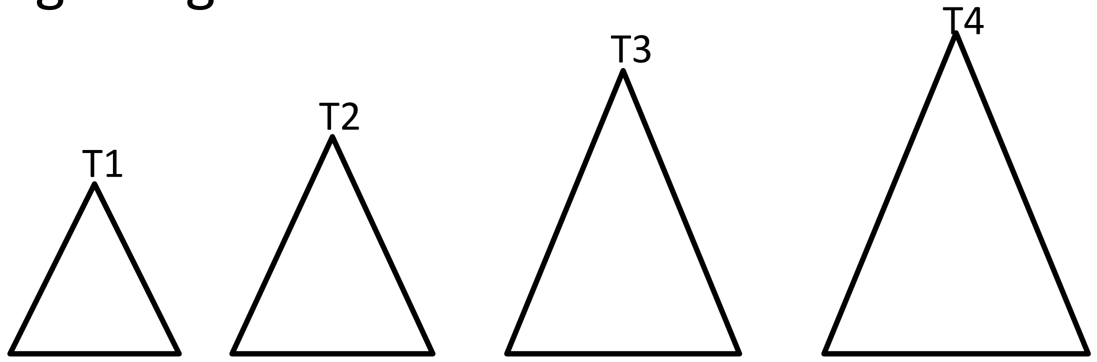
- Bagging using ASHTs of different sizes



  - Intuition:
    - Smaller trees adapt more quickly to changes
    - Larger trees perform better during periods with no or little change because they are built over more data
  - Trees limited to size s will be reset about twice as often as trees with a size limit of 2s.
    - This creates a set of different reset-speeds for an ensemble of such trees, and therefore a subset of trees that are a good approximation for the current rate of change.

Ensemble of Adaptive Size Hoeffding Trees

- Bagging using ASHTs of different sizes



  - The max allowed size for the nth ASHT tree is twice the max allowed size for the (n-1)th tree.
  - Each tree has a weight proportional to the inverse of the square of its error
  - The size of the first tree is 2
  - Goal: improve bagging performance by increasing tree diversity

Outline

- Introduction
- Data stream classification
- Data stream classifiers
- Data stream classifiers evaluation
- Things you should know
- Homework/tutorial

(batch) Classifier evaluation

- The quality of a classifier is evaluated over a *test set*, different from the training set
- For each instance in the test set, we know its true class label
- Compare the predicted class (by some classifier) with the true class of the test instances
- Terminology
  - Positive tuples: tuples of the main class of interest
  - Negative tuples: all other tuples

- A useful tool for analyzing how well a classifier performs is the *confusion matrix*
- For an m-class problem, the matrix is of size m x m
- An example of a matrix for a 2-class problem:

Predicted class

Actual class

	C_1	**C_2**	**totals**
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P’	N’	

(batch) Classifier evaluation measures

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P'	N'	

- Accuracy/ Recognition rate:
  - % of test set instances correctly classified

$$accuracy(M) = \frac{TP + TN}{P + N}$$

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	
buy_computer = no	412	2588	3000	
total	7366	2634	10000	95.42

- Error rate/ Missclassification rate: error_rate(M)=1-accuracy(M)

$$accuracy(M) = \frac{FP + FN}{P + N}$$

- More effective when the class distribution is relatively balanced
  - Check Lecture 4, KDD I for more evaluation measures also if classes are imbalanced!

(batch) Classifier evaluation methods

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (~2/3) for model construction, Test set (~1/3) for evaluation
- Cross-validation (k-fold cross validation, k = 10 usually)
  - Randomly partition the data into k mutually exclusive subsets $D_1, \dots, D_k$ each approximately equal size
  - Training and testing is performed k times
    - At the i-th iteration, use D_i as test set and others as training set
  - Accuracy is the avg accuracy over all iterations
- Bootstrap: Samples the given training data uniformly with replacement
  - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Check Lecture 4, KDD I for more evaluation methods, their pros and cons!

Evaluation methods for streams

- Holdout evaluation
  - 2 separate datasets for training (~70% - 80% of the dataset) and testing (~20%-30% of the dataset)
  - Train model on training set
  - Test model on test set
    - Static test set!!!

- Prequential evaluation (Interleaved test-then-train)
  - One dataset for training and testing
  - Models are first tested then trained in each instance
    - Test set is dynamic!!!

Evaluation measures for streams

- Accuracy

- Kappa measure
  - normalizes the accuracy of a classifier p_0 by that of a chance predictor p_c
  - appropriate for imbalanced classes

$$k = \frac{p_0 - p_c}{1 - p_c}$$

$$p_c = \sum_{i=1}^{k} \frac{r_i}{n} \frac{c_i}{n}$$

r_i/n: predicted frequency of class i
c_i/n: actual frequency of class i

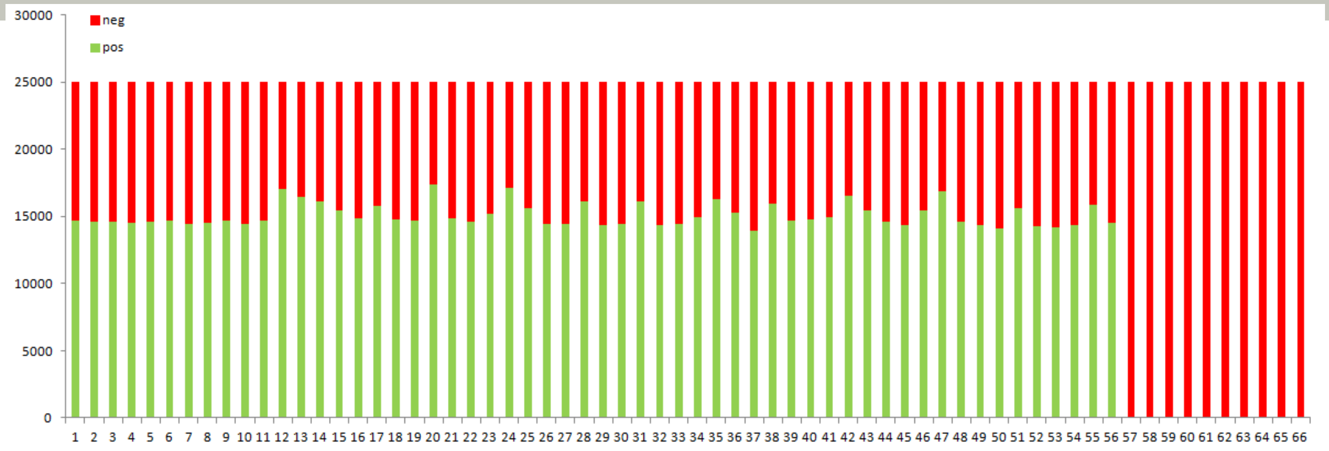
Kappa value	Classifier‘s performance
0%-20%	bad
21%-40%	fair
41%60%	moderate
61%-80%	substantial
81%-100%	(almost) perfect

- Both measures are computed based on the most recent samples through some
  - sliding window
  - fading function

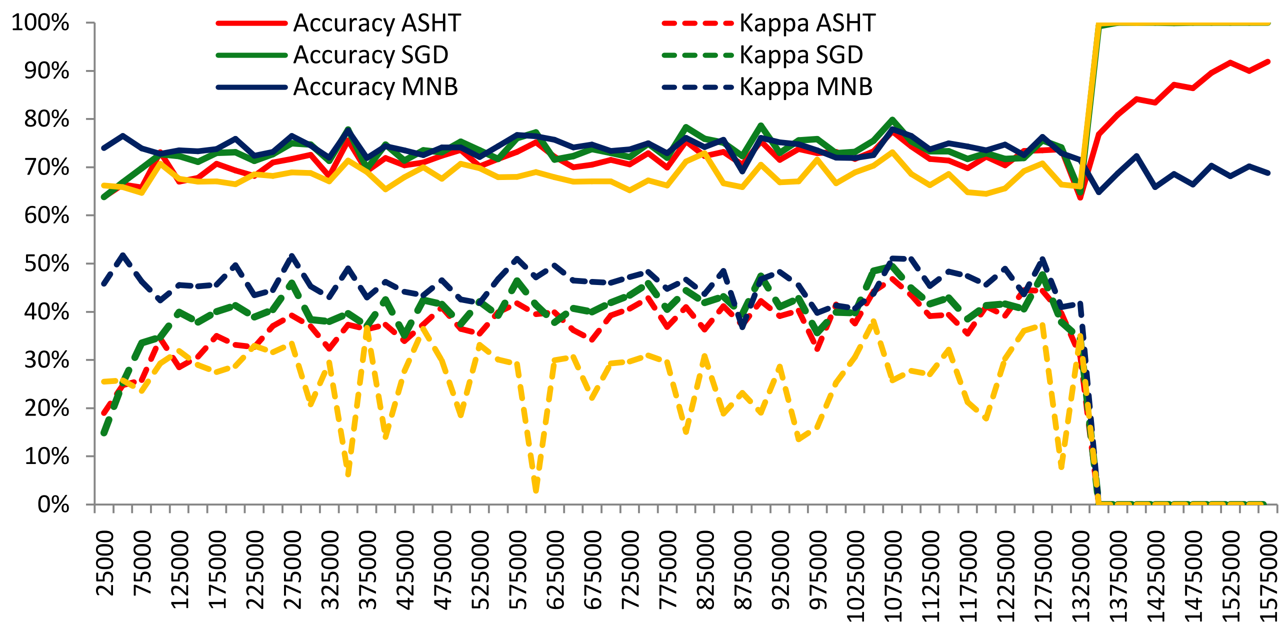
DATABASE SYSTEMS GROUP

Accuracy vs Kappa

Experiments from the BA of Alina's Sinelnikova , "*Sentiment analysis in the Twitter stream*", LMU 2012.



Evolving data distribution



Prequential evaluation

DATABASE SYSTEMS GROUP

Outline

- Introduction
- Data stream classification
- Data stream classifiers
- Data stream classifiers evaluation
- Things you should know
- Homework/tutorial

Things you should know

- Stream classification goals and challenges
- Hoeffding trees
  - Hoeffding bound
  - Adaptive size Hoeffding tree
- Ensembles of classifiers and their application to streams
  - Ensembles basic idea
  - Ensemble of adaptive size Hoeffding trees
- Evaluation methods for streams
  - Holdout
  - Prequential
- Evaluation measures for streams
  - Accuracy
  - Kappa statistic

Resources

- J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall/CRC, 2010.
- New ensemble methods for evolving data streams, Bifet et al, KDD 2009
- State of the art in data stream mining, Gaber and Gamma, ECML/PKDD 2007
- Mining High Speed Data Streams, Domingos and Hulten, KDD 2000

Homework/Tutorial

- Tutorial:
  - Thursday 6.12.2012 on stream classification
- Homework:
  - Try some stream classification algorithm in the MOA framework (Weka extension to streams): http://moa.cms.waikato.ac.nz/