Data Mining II

Online Learning
Overview

- Data Streams
  - In comparison to offline data
  - Theorem of Concept Drifts
  - Batch Learning

- Online Learning
  - Clustering
  - Classification

- Active Learning
Introduction

- Data is growing all the time
- Every second new data is generated
- So far prediction of the models were based on static environment:
  - Customer and shopping behavior stays the same (no matter what season and so on)
  - Distribution of characteristics (e.g. 20% male, 80% female) is static

→ Data is not static, it is a stream which continuously comes with new examples
Applications with Streams

- Sensor data (industry and cities)
  - Oil & gas platform
  - Traffic surveillance

- Telecom-Data
  - Traffic in the network
  - Based on time-zones
  - Based on holidays

- Social Networks
  - Twitter, Facebook, Google+

- Marketing
  - Shopping platforms
  - Prediction of behaviors
Wrap up what we did so far

– Fixed/static set of examples for training & testing
– Direct/random access to the data
– Example set is known:
  • Number of attributes
  • Attribute distribution
  • Number of missing values
  • Examples for training and testing
  • Ratio between classes
  • and others
Data Streams

- No fixed/static set of examples but continues stream of data
- “End” of the data is not known
- Only sequential access to the data (in contrast to direct/random access)
- Each period new data becomes available with potentially:
  - New examples
  - New attributes
  - New characteristics of attributes
  - New distributions
  - ...
- Data rate (number of new examples per time period) can vary strongly
- We potentially get feedback for the examples short after receiving them
Challenges: Data Streams

- With the increasing volume multi-pass processing might not be possible any more
  - In most cases, an example can only be processed once
  - Stream mining algorithms mostly implement the „one pass“ strategies

- Stream data tend to evolve over time
  - Referred as „temporal locality“
  - Direct transformation of one pass mining algorithms might not work
  - Stream mining algorithms must be able to adjust on changes
Example

- Find best attribute split for Decision Trees (HUNTS algorithm)
  - Calculate each possible split and use split with best impurity

- Questions:
  - Which examples should we include in this calculation?
  - Can we ensure that this is really the best split?
  - Who long is this decision valid? What is the accuracy of our model after 1, 2 or 10 new periods?
Stream Mining Principles

1. Process an example at a time and inspect it only once (at most)

2. Use limited amounts of memory

3. Work in limited amount of time

4. Be ready to predict at any point
Overview: Stream Mining

- Change Detection in Data Streams
- Sliding Window Computation in Data Streams
- Data Stream Clustering
- Data Stream Classification
- Frequent Pattern Mining
- Indexing Data Streams

And there are many more …
Concept Drift

- Example: Learning buy decisions based on e-commerce log data.
  - Our learned model tells us, that in 50% of the cases a new customer is interested in short trousers → we show short trousers on the landing page
  - BUT: but current sales data states, that we do not sell any short trousers
  - What happened? → Winter is coming!

- This effect is called **concept drift**

- Concept drift means the statistical properties of the target variable, which we are trying to predict changes over time in unforeseen ways.
Concept Drift

- Concept drift makes the handling of data streams complex
- If there is no drift, we could simply use a static model based on a comprehensive set of examples
- Whenever a concept drift is detected we need to adapt our data mining algorithm to reflect this drift
Main properties of an optimal concept drift detection and prediction system:

- High accuracy in the prediction → We want the best possible results
- Low mean time to detection (MTD), false positive rate and missed detection rate → We want the detection fast, with no missed alarms!
- Low computational costs: minimum space and time → We cannot dedicate too much resources to this process
- Theoretical guarantees → We need to rely on this, to a certain extend
- No parameters needed → We cannot “tune” this detection!
Can we adapt our algorithms to data streams?

- What do we have:
  - Examples from a stream
  - Feedback right after we see the example (or at least after a certain period of time/examples)

- What do we want:
  - A prediction for each example we see

- Idea: We collect a sufficient amount of data from the stream, than learn a model and apply this model to the rest of the data coming from the stream

- Drawback: We might miss some parts of the data or some changes (concept drifts) and we might be late (too slow)
Continuous Batch Learning

- **Improvement:** We relearn the model after a fixed time period or based on resource limitations

- **Benefits:**
  - We will not miss parts of the data
  - Model can be (to some extend) learned in the background and replace old model as soon as it is computed

- **Drawbacks:**
  - We need to store the seen data somehow, which can be costly and time consuming
  - We always can only use that amount of data, which we can process in a given time
  - We might miss a change happening during two learning phases
Batch Learning In RapidMiner

- Simulation of batches
- For each month, we learn a model with the past month (0 to t-1) and apply it to the upcoming month (t)
Sliding Window

- Based on the assumption, that recent data in data streams is more important than all data, or data which is further away
- We only use the data which is available in the window to calculate a model
- Major challenge: Find/compute the optimal window size based on given resources and task:
  - Memory/Resources
  - Time
  - Data rate
- Benefits:
  - Within the window we can maintain basic statistics
  - We forget old/outdated data
Sliding Window Evaluation in RapidMiner

- Time Series Extension offers a Sliding Window Evaluator

- Performance (Forecasting)

- Window Size (Train & Test)

- Timespan to example which should be predicted (horizon)
Sliding Window Evaluation in RapidMiner

- **Process:**
  - From first example to last example (temporal order)
    - Select window of training data
    - Train for an example in the future (horizon)
    - Test data using data from a window of test data
    - Evaluate test data
  - Calculate average performance

- **Drawbacks:**
  - Ultimate Batch Learning, as we train a new model after each new example
  - Time consuming and costly
  - Fixed window size
Wrap Up

- Data Streams and their properties
- Concept drifts
- Adaption of classic offline learning for data streams
  - Batch learning
  - Sliding Window technique
Online Machine Learning

- Also called stream machine learning
  - Do not mix it up with e-Learning
- Similar to offline learning
- Restrictions from Data Streams:
  - Massive amount of data
  - Limited time to handle data
Online Machine Learning

- General process flow
  1. Receive example $e$
  2. Predict label for $e$
  3. Receive true label for $e$

- Depending on the used model, after receiving the true label of $e$, the information can be incorporated to improve the model

- Questions:
  1. Should we modify our model to fit the data characteristics?
  2. What kind of modification should be performed?
Online Machine Learning

- **Benefits:**
  - Short feedback loops can be incorporated
  - Is able to handle stream data
  - Able to be trained and adapted for difficult situation (concept shifts)
  - Only need to store one example in RAM (and the model(s))

- **Drawbacks:**
  - Online learning approaches need feedback
  - Feedback might be not accurate
    - We do not create hand-labelled data on the fly
Wrap Up: Clustering

- **Definition**: Clustering is the distribution of a set of instances of examples into non-known groups according to some common relations or affinities

- Examples:
  - Social Network Communities
  - Market segmentation of customers

- Known Algorithms:
  - k-means/k-medoids
  - Hierarchical Clustering
  - DBScan (Density based)
Stream Clustering: \textit{leader}

- Spath (1980)
- Uses user-defined distance threshold

\begin{algorithm}
\textbf{Data:} $X$: Sequence of Examples $x_i$, $\delta$: Control Distance parameter.
\textbf{Result:} Centroids of the k Clusters
\begin{algorithmic}
\STATE Initialize the set of centroids $C = x_1$
\STATE \textbf{foreach} $x_i \in X$ \textbf{do}
\STATE \hspace{1em} Find the cluster $C_r$ whose center is close to $x_i$
\STATE \hspace{2em} \textbf{if} $d(x_i, C_r) < \delta$ \textbf{then}
\STATE \hspace{3em} $C_r = C_r \cup x_i$
\STATE \hspace{1em} \textbf{end}
\STATE \textbf{end}
\STATE Remove Points from the buffer
\end{algorithmic}
\end{algorithm}

- Performance heavily depends on the order of the examples and a correct guess of the distance threshold
Stream Clustering: k-means/k-medoids

- Farnstrom et al. (2000) and Guha et al. (2003)
- Batch-based approach

---

```
Data: S: A Sequence of Examples, k: Number of desired Clusters
Result: Centroids of the k Clusters

begin
    Randomly initialize cluster means;
    Each cluster has a discard set that keeps track of the sufficient statistics.
    while TRUE do
        Fill the buffer with examples;
        Execute iterations of k-means on points and discard set in the buffer, until convergence;
        /* For this clustering, each discard set is treated like a regular point weighted with the number of points in the discard set. */
    end
    foreach group do
        update sufficient statistics of the discard set with the examples assigned to that group
    end
    Remove Points from the buffer
end
```
Stream Clustering: BIRCH

- Zhang et al. (1996)
- Balanced Iterative Reducing and Clustering using Hierarchies
- Builds up a hierarchical structure, the CF-Tree
  - CF = Clustering Feature, with sufficient statistics to describe a set of data points, and compress all information of the CFs below in the tree
- Input: branch factor B and maximum distance T in the leaf CF
- Works only with continuous attributes
- Time and Memory sensible
  - Tries to find best groups for given resources
- Operates in 2 phases:
  - On-line: Compression of data in-memory
  - Off-line: Using arbitrary clustering algorithm
On Demand Mining using Micro-clustering

- Extending CF with Timestamp (CFT) also called Micro-Cluster (MC)
  - Example definition: \( MC = (WSS, w, t_c) \)
    - \( WSS \) = Weighted Square Sum
    - \( w \) = weight of MC
    - \( t_c \) = creation time of MC

- Idea: Dividing the mining process into two layers
  - On-line layer: generation of a local model (micro-clusters) in-memory, data points are aggregated based on statistical similarities
  - Off-line layer: generation of a global model based on micro-clusters, data points are not taken as single points, but a MC to perform the algorithm

- Powerful idea for a large set of stream mining problems
  - Data is aggregated on the fly
  - Amount of single data clusters is reduced to a manageable size
  - Applying of arbitrary (sometimes customized) algorithms
Wrap Up: Classification

- **Definition**: Methods and criteria to distinguish examples into pre-defined classes based on models.

- **Examples**:
  - Detection of fraud within network traffic (nominal classification)
  - Prediction of DAX closing values (regression)

- **Known Algorithms (excerpt)**:
  - K-NN
  - Naïve Bayes
  - Decision Trees
  - Support Vector Machines
  - Neural Networks
  - RIPPER
Stream Classification: Ensembles

- Wang et al. (2003)

- Idea: Learn individual base classifiers from relative small subset of the data (sequentially read in blocks) and use ensemble (e.g. voting). Outdated classifiers are replaced with better performing “new” classifier

```
while more data points are available do
    read \( d \) points, create training set \( D \)
    build classifier \( C_i \) using \( D \)
    evaluate classifier \( C_{i-1} \) on \( D \)
    evaluate all classifiers in ensemble \( E \) on \( D \)
    if \( E \) not full then
        insert \( C_{i-1} \)
    else
        if \( \text{Quality}(C_{i-1}) > \text{Quality}(E_j) \) for some \( j \) then
            replace \( E_j \) with \( C_{i-1} \)
        end
    end
end
```
Stream Classification: ANNCAD

- Law et al. (2005)
- **Adaptive Nearest Neighbor Classification for Data Streams**
- Haar Wavelets Transformation for multi-resolution data representation

\[(8+9+10+0)/4\]

Pictures by Yan-Nei Law & Carlo Zaniolo
University of California
Stream Classification: ANNCAD

- On each level labelled with major class (C1-C2 > 80%) of previous level

1. For classification, start on lowest resolution, whenever the classification is unclear go one level up.
2. If no level finds a decision, compute distance to neighboring blocks

- Labelling is recalculated with new data point
- Concept drift is faced by putting exponential weights on old data
Stream Classification: ANNCAD

- Classification Example

(a) Classified block
   \rightarrow Label class I

(b) Unclassified block, go to next level.

(c) Block with tag “M”, go back to prev. level.

Compute the distance between the test point and the center of every nonempty neighboring block.

The combined classifier over multiple levels
Stream Classification: Hoeffding Trees

- Domingos and Hulten (2000)

- Learning is based on Very Fast Decision Tree Algorithms (VFDT)
  - Recursively replacing leaves by decision nodes (splits)
  - Sufficient statistics about attribute values are stored in the leaves
  - Those statistics are necessary to evaluate the split-tests
  - Before installing a split (instead of a leaf), sufficient examples are collected
  - Only split condition is stored for a decision node

- Named after **hoeffding bounds**, Hoeffding (1963)
  - Used to determine how many examples are sufficient, before installing the split

- Fixed time for learning (each example needs a fixed amount of time to be incorporated in the current model)

- Resulting tree is almost similar to a tree build with a batch approach
Stream Classification: Hoeffding Trees

– Hoeffding Tree learning algorithm

1: Let $HT$ be a tree with a single leaf (the root)
2: for all training examples do
3:   Sort example into leaf $l$ using $HT$
4:   Update sufficient statistics in $l$
5:   Increment $n_l$, the number of examples seen at $l$
6:   if $n_l \mod n_{\text{mix}} = 0$ and examples seen at $l$ not all of same class then
7:     Compute $\overline{G}_l(X_i)$ for each attribute
8:     Let $X_a$ be attribute with highest $\overline{G}_l$
9:     Let $X_b$ be attribute with second-highest $\overline{G}_l$
10:    Compute Hoeffding bound $\epsilon = \sqrt{\frac{2 \ln(1/\delta)}{2n_l}}$
11:   if $X_i \neq X_b$ and $(\overline{G}_l(X_e) - \overline{G}_l(X_b)) > \epsilon$ or $\epsilon < \tau$ then
12:     Replace $l$ with an internal node that splits on $X_a$
13:     for all branches of the split do
14:       Add a new leaf with initialized sufficient statistics
15:     end for
16:   end if
17: end if
18: end for

– Question: How do we classify new examples with this tree when the leaves are mixed?
Stream Classification: Hoeffding Trees

- Hoeffding Trees might be *incomplete* within the leaves as a large number of (hundreds or thousand) examples are necessary to fulfill the hoeffding boundary

- Option: Major class decision
  - Each example is classified based on the most dominant class in this leaf

- Better Option: Use Naïve Bayes for each leaf
  - Number of examples in the leaf is manageable
  - Naïve Bayes takes into account:
    - Prior class distribution
    - Conditional probabilities o attribute-values for each class
Stream Classification: On Demand Classification

- Aggarwal et al. (2004)
- Two component approach (similar to BIRCH, using MCs)
  - Component 1: continuously stores summarized statistics about data stream
  - Component 2: uses summary statistics to perform classification, by assigning each MC (from component 1) a class

- Classification model is defined over a time horizon:
  - Minor concept drift: large time horizon to ensure robustness
  - Major concept drift: small time horizon is required
  - Remember: Each MC has a timestamp of its creation.
Online Learning in RapidMiner

- We can also simulate stream learning with the same approach as batch but this would be time consuming (relearning the offline approach all the time)

- There is also no extension for this task available (as far as I know) which also incorporates the presented online algorithms

- In the exercise we will use MOA (Massive Online Analysis) which is capable of handling stream data
Massive Online Analysis (MOA)

- Data Stream Analysis in Real Time
- Java Based Open Source Framework of the University of Waikato
- [http://moa.cms.waikato.ac.nz/](http://moa.cms.waikato.ac.nz/)

- Scalable Version:
  - SAMOA (scalable advanced massive online analysis)
Evaluation of Stream Mining Learning

- Error estimation:
  - Hold-out or Prequential

- Evaluation performance measures:
  - Accuracy or k-statistic
Error Estimation

- Data available for testing:
  - Holdout an independent test set
  - Apply the current decision model to the test set, at regular time intervals
  - The loss estimated in the hold out is an unbiased estimator

- No data available for testing:
  - The error of the model is computed from the sequences of examples
  - For each example in the stream, the actual model makes a prediction, and then uses it to update the model

- Hold-out is more accurate, but needs data for testing
  - Use prequential to approximate hold-out
  - Estimate accuracy using sliding windows or fading factors
Performance Measure

- Similar to static classification we use accuracy
- Beside this we can also use:
  - Accuracy: \( \frac{TP + TN}{TP + TN + FP + FN} = 85\% \)
  - Arithmetic mean: \( \frac{\frac{TP}{TP + FP} + \frac{TN}{TN + FN}}{2} = 74.59 \)
  - Geometric mean: \( \sqrt{\frac{TP \times TN}{(TP + FP) \times (TN + FN)}} = 72.90\% \)

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Predicted Class-</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Class+</td>
<td>75</td>
<td>8</td>
</tr>
<tr>
<td>Correct Class-</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>82</td>
<td>18</td>
</tr>
</tbody>
</table>

Table: Simple confusion matrix example
Kappa Statistic

- $p_0$: classifiers prequential accuracy
- $p_c$: probability that a chance classifier makes a correct prediction
- $k$ statistic: $k = \frac{p_0 - p_c}{1 - p_c}$
  - $k = 1$ if the classifier is always correct
  - $k = 0$ if the prediction coincide with the correct ones as often as those of the chance classifier

- Forgetting mechanism for estimating prequential kappa
  - Sliding window of size $w$ with the most recent observations
### Cost Evaluation Example

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier A</td>
<td>70%</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Classifier B</td>
<td>80%</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

Which classifier is performing better?
## Cost Evaluation Example

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Time</th>
<th>Memory</th>
<th>RAM-Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier A</td>
<td>70%</td>
<td>100</td>
<td>20</td>
<td>2,000</td>
</tr>
<tr>
<td>Classifier B</td>
<td>80%</td>
<td>20</td>
<td>40</td>
<td>800</td>
</tr>
</tbody>
</table>

Which classifier is performing better?
Distributed Streaming

- When data becomes too big for a single machine

- Frameworks:
  - Hadoop, Storm, S4

- Hadoop:
  - Apache Mahout (Data Mining Open Source)
  - PIG (Similar to SQL – Data-centric)

- Storm
  - Used by Twitter
  - Stream based hadoop-like infrastructure

- S4
  - Apache incubator
  - Batch oriented infrastructure
Wrap up: Online Learning

- Stream Clustering
  - leader
  - K-means
  - BIRCH

- Stream Classification
  - Ensembles
  - ANNCAD
  - Hoeffding Trees
  - On Demand

- Evaluation of Data Stream Classification
Further Reading

- Knowledge Discovery from Data Streams by Joao Gama (CRC Press)

- Data Streams: Models and Algorithms edited by Charu C. Aggarwal (Springer)
Questions?
Active Learning

- Motivation: Labelling training data is costly (time and resources)

- Definition: Process of guiding the sampling process by querying for certain types of instances based upon the data that we have seen so far

- In contrast to random selection of training data, active learning iteratively picks examples that are predicted to be maximally informative for the given target
Active Learning Scenarios

- There are 3 different major active learning scenarios
  1. Membership query synthesis: Learner requests labels for instances, generated by the learner itself
  2. Stream-Based Selective Sampling: Learner decides whether the label for the current instance is required (and will be queried) or if it is left out (discard)
  3. Pool-based Sampling: Learner picks the best instances from a pool which should be queried
Active Learning Query Strategies

- **Uncertainty Sampling:**
  - Instance is selected with lowest confidence on the label

- **Query-By-Committee:**
  - Instance is selected where a committee of models (e.g. classifiers) disagree most

- **Expected Model Change:**
  - Instance is selected, which imparts the greatest change in the current model, if we would know the label

- **Expected Error Reduction**
  - Instance is selected, on the likelihood to reduce the generalization error

- **Variance Reduction**
  - Instance is selected, on the likelihood to reduce the variance of the output
Wrap up: Active Learning

- Small set of selected examples is used to train
- Cost-efficient
- Different selection strategies
Questions?