دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

Discretization Methods

Sina Malakouti

Nima Tavassoli

What is Discretization

- Discrete values are intervals in a continuous spectrum of values.
- A process of quantizing continuous attributes.

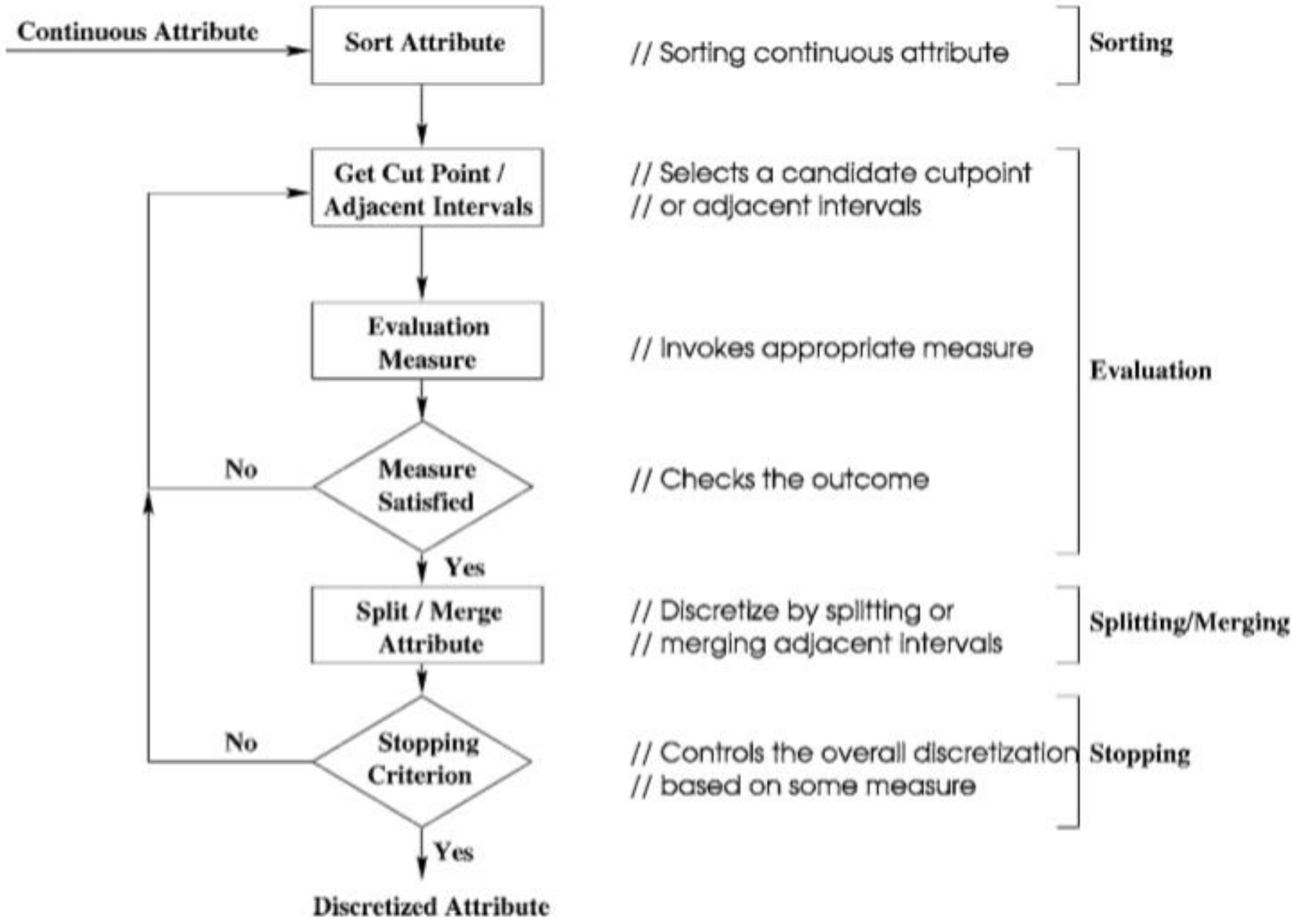
Why Discretization?

- Pre-Processing
- Easier to use
- Human interpretability
- Faster and more accurate models

Terms and Notations

- **Feature** or “Attribute” or “Variable” refers to an aspect of the data. Usually before collecting data, features are specified or chosen. Features can be discrete, continuous, or nominal.
- **Instance** or “Tuple” or “Record” or “Data point” refers to a single collection of feature values for all features
- **Cut-point** refers to a real value within the range of continuous values that divides the range into two intervals, one interval is less than or equal to the cut-point and the other interval is greater than that.
- **Boundary** refers to the candidate cut-points.

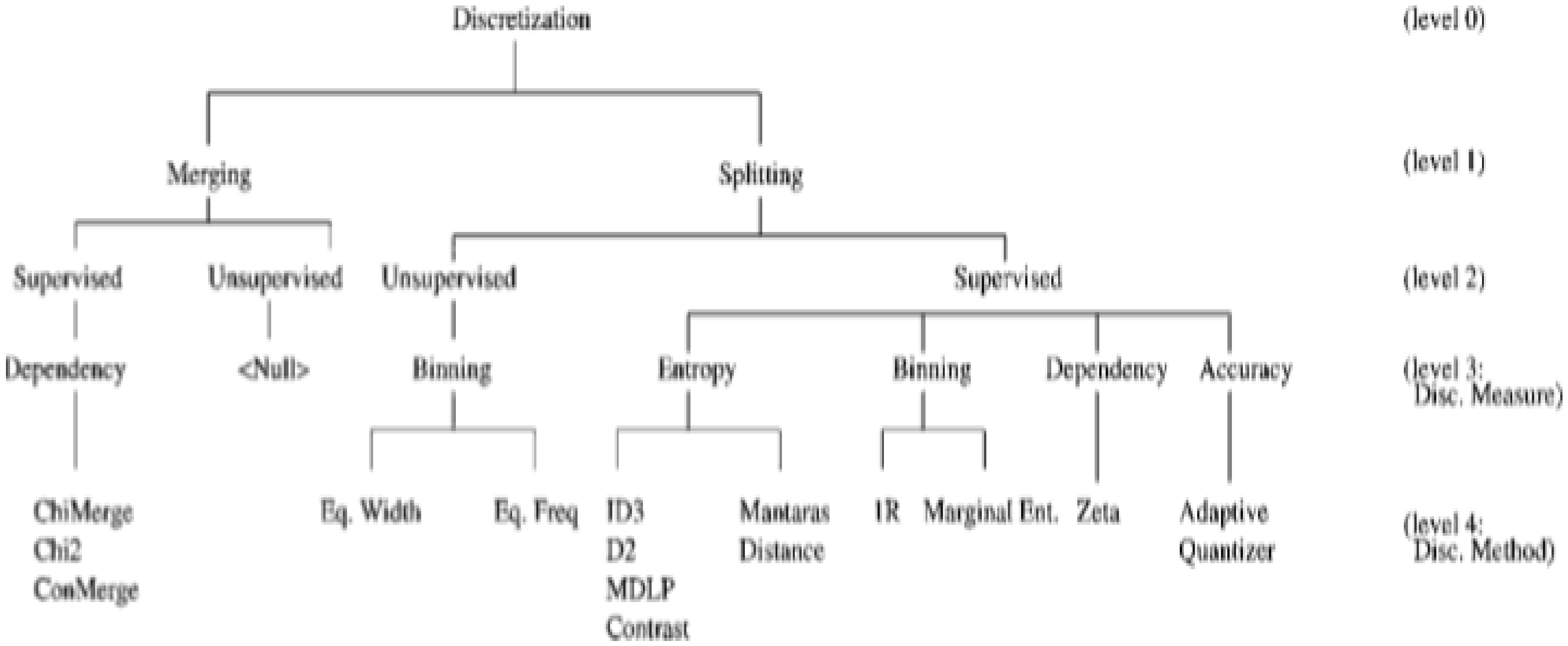
Discretization process



Types

- **Unsupervised Methods**
  - Class labels are not encountered
  - Equal width intervals, arbitrarily cut-points, quantile points (IDA)
- **Supervised Methods**
  - Class labels are take into account
  - Same class labeled intervals are desired
  - MDLPC, Chi-merge, FUSINTER, PiD, LOFD

Types – Cont.'


Discretization
(level 0)
Merging
Splitting
(level 1)
Supervised
Unsupervised
Unsupervised
Supervised
(level 2)
Dependency
<Null>
Binning
Entropy
Binning
Dependency
Accuracy
(level 3: Disc. Measure)
ChiMerge
Chi2
ConMerge
Eq. Width
Eq. Freq
ID3
D2
MDLP
Contrast
Mantaras Distance
1R
Marginal Ent.
Zeta
Adaptive Quantizer
(level 4: Disc. Method)

Types – Cont.'

Merging Algorithm

S = *Sorted values of feature* f

```
Merging(S){
    if StoppingCriterion() == SATISFIED
        Return
    T = GetBestAdjacentIntervals(S)
    S = MergeAdjacentIntervals(S, T)
    Merging(S)
}
```

Splitting Algorithm

S = *Sorted values of feature* f

```
Splitting(S){
    if StoppingCriterion() == SATISFIED
        Return
    T = GetBestSplitPoint(S)
    S1 = GetLeftPart(S, T)
    S2 = GetRightPart(S, T)
    Splitting(S1)
    Splitting(S2)
}
```

MDLPC

- Minimum Description Length Principal Cut
- Fayad and Irani (1993)
- Top-Down
- Entropy Based

MDLPC- Steps

1. Sort examples in an ascending manner
2. Each runs of a same class forms an interval
3. The discretization points are necessarily taken from the boundaries
4. Best bi-partition split
5. Stop when no improvement is feasible

MDLPC- Criterion

- $\psi(d) = Gain(d) - \frac{\log_2(n-1)}{n} - \frac{\delta(d)}{n}$

  where, n training size and d is boundary

we choose the discretization point d^* that checks :

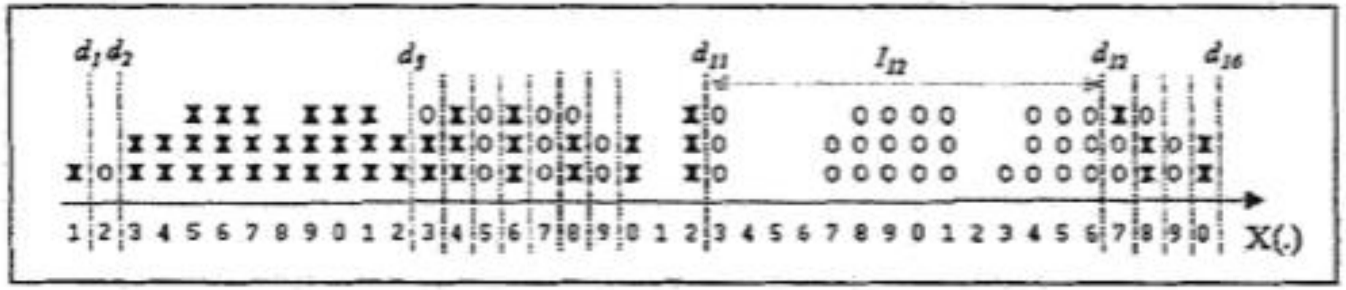
$$\begin{cases} d^* = \arg\max_d[\Psi(d)] \\ \Psi(d^*) > 0 \end{cases}$$

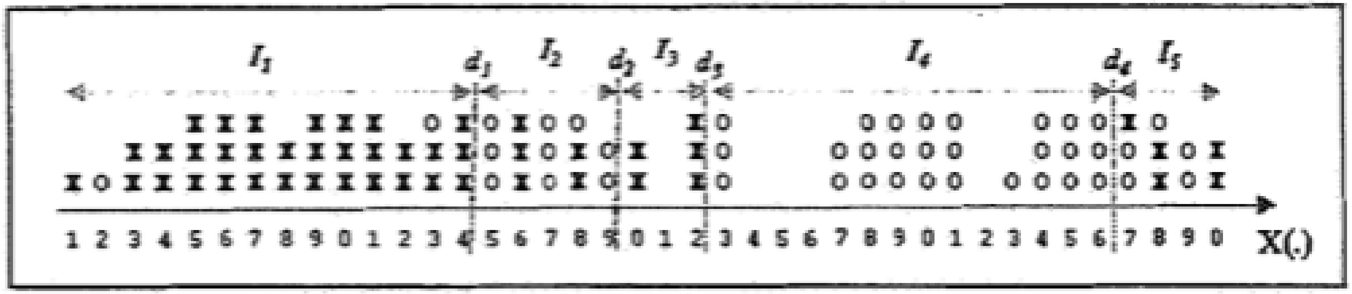
Notations are:

- $h(\Omega) = -\sum_{i=1}^{m} \frac{n_{i.}}{n} Log_2 \frac{n_{i.}}{n}$, the Shannon entropy;
- $h(\Omega_j) = -\sum_{i=1}^{m} \frac{n_{ij}}{n_{.j}} Log_2 \frac{n_{ij}}{n_{.j}}$, the conditional entropy;

- $Gain(d_t) = h(\Omega) - h(\Omega_j)$, the entropy gain criterion;
- $\delta(d) = log_2(3^m - 2) - mh(\Omega) + \sum_{j=1}^{2} m_j h(\Omega_j)$.

MDLPC





Chi-Merge

- Kerber (1992)
- Bottom-up
- Entropy based

Chi-Merge - Steps

1. Sort examples in ascending manner
2. Each value forms an interval
3. To each interval I_j is associated a distribution T_j
4. Merge the pair of adjacent intervals
5. Stop when no improvement is possible

Chi-Merge -Criterion

- Statistical Criterion chi-squared test

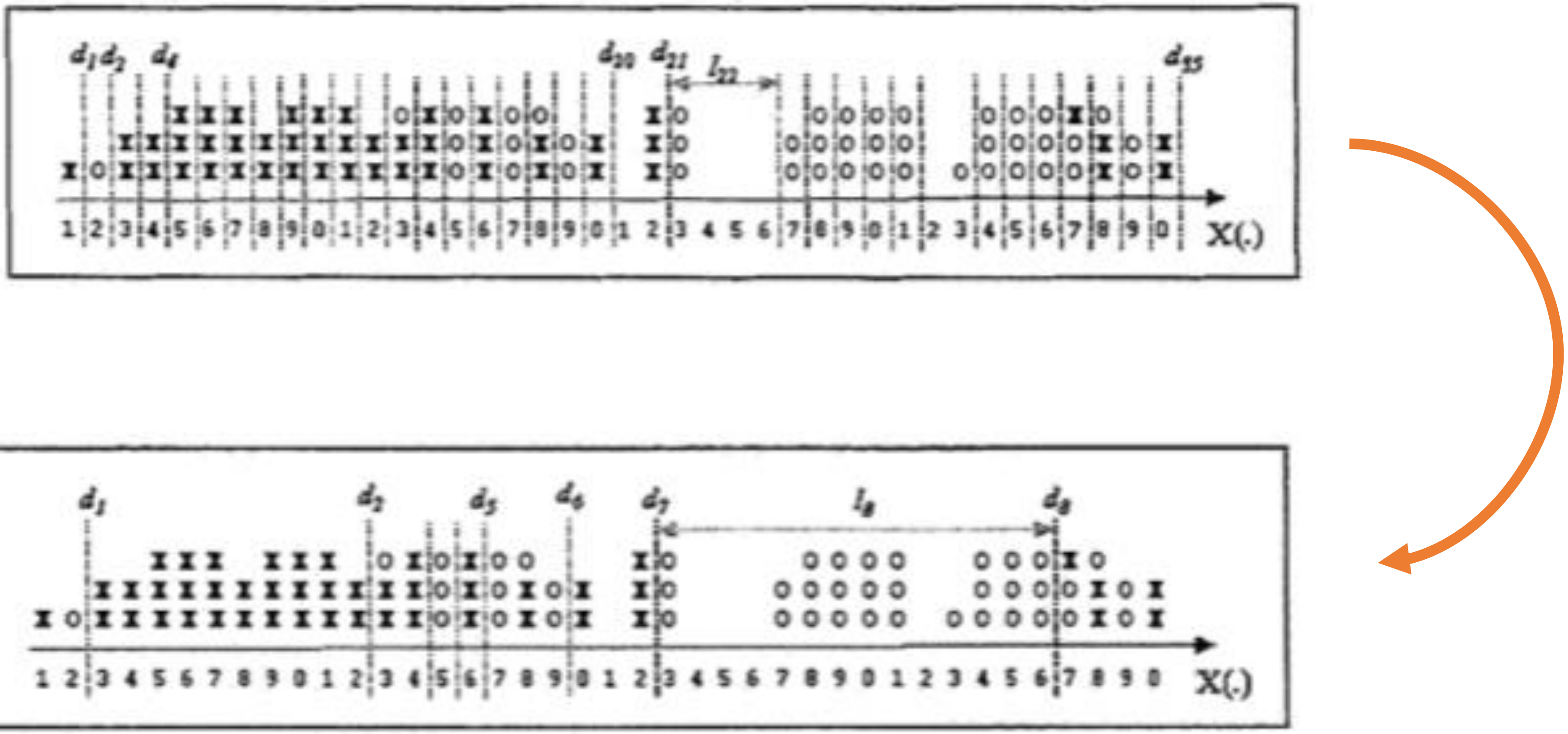
$$\chi^2(T_q, T_{(q+1)}) = \sum_{i=1}^{m} \sum_{j=q}^{q+1} \frac{(n_{ij} - n_{.j} \sum_{k=q}^{q+1} n_{ik})^2}{n_{.j} \sum_{k=q}^{q+1} n_{ik}}$$

- Merge the pair of adjacent intervals that gives the smallest value of chi squared and checks the following:

$$\chi^2(T_q, T_{(q+1)}) < \chi^2(\alpha, m-1)$$

Where α is type I error and m -1 is degrees of freedom

Chi-Merge – Cont.’



FUSINTER

- Zighed, Loudcher (1998)
- Bottom-up
- Entropy based
- Main characteristic: sample size sensitiveness
- In contrast to Chi-Merge, find the partition with optimized measure
- Avoid thin partitioninig

FUSINTER- Steps

1. Sort examples in an ascending manner
2. Each runs of a same class forms an interval
3. The discretization points are necessarily taken from the boundaries
4. Merges two adjacent intervals
5. Stop when no improvement is feasible

$$\varphi(T)-\varphi(\ldots,\{T_j+T_{(j+1)}\},\ldots) = Max_{i=1}^{k-1}\left(\varphi(T)-\varphi(\ldots,T_i+T_{(i+1)},\ldots)\right)$$

7. : If

$$\varphi(T)-\varphi(T_1,\ldots,T_j+T_{(j+1)},\ldots,T_k) > 0$$

FUSINTER- Criterion

- Merges two adjacent intervals whose merging improve the criterion
- **Axioms**
    1. Minimality
    2. Maximality
    3. Sensitiveness to the sample size
    4. Symmetry
    5. Merging
    6. Independence

FUSINTER- Criterion Cont.'

- based on Shannon's entropy

$$\varphi_1(T) = \sum_{j=1}^{k} \alpha \frac{n_{.j}}{n} \left(- \sum_{i=1}^{m} \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} log_2 \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \right) + (1 - \alpha) \frac{m\lambda}{n.j}$$

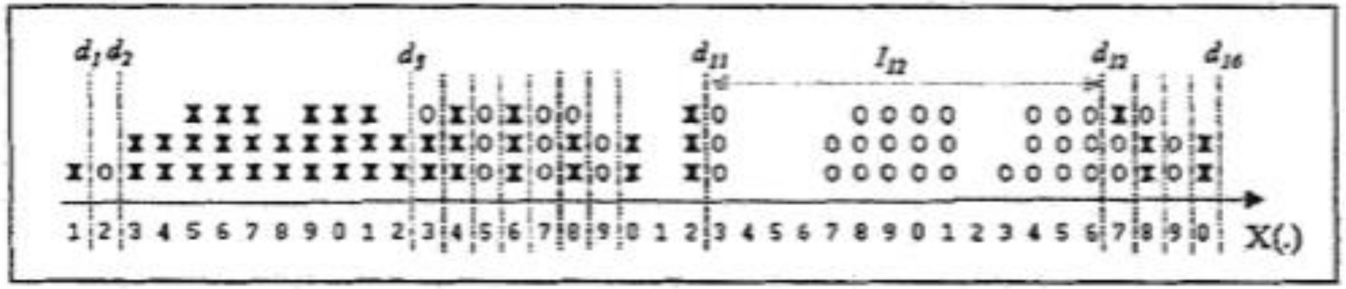
- Based on quadratic entropy

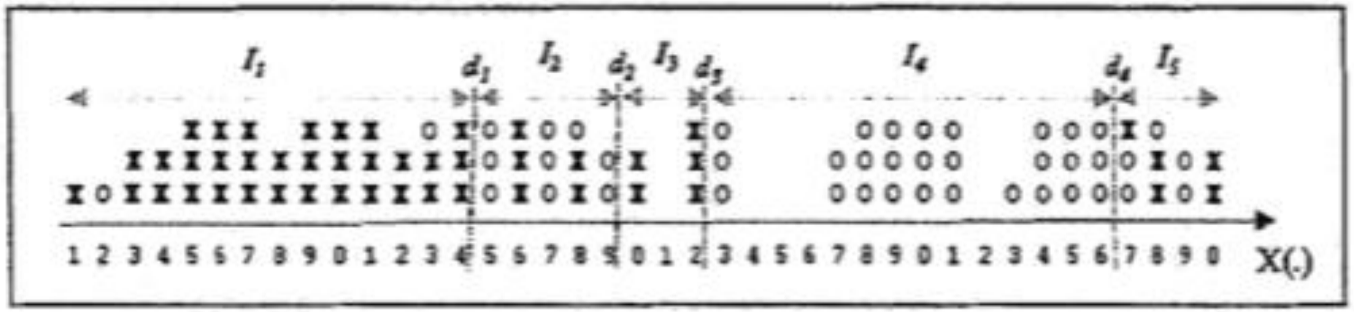
$$\begin{aligned} \varphi_2(T) &= \sum_{j=1}^{k} \alpha \frac{n_{.j}}{n} \left(\sum_{i=1}^{m} \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \left(1 - \frac{n_{ij} + \lambda}{n_{.j} + m\lambda} \right) \right) + (1 - \alpha) \frac{m\lambda}{n.j} \\ &= \sum_{j=1}^{k} \alpha H_j(h, \lambda) + (1 - \alpha) \frac{m\lambda}{n_{.j}} \end{aligned}$$

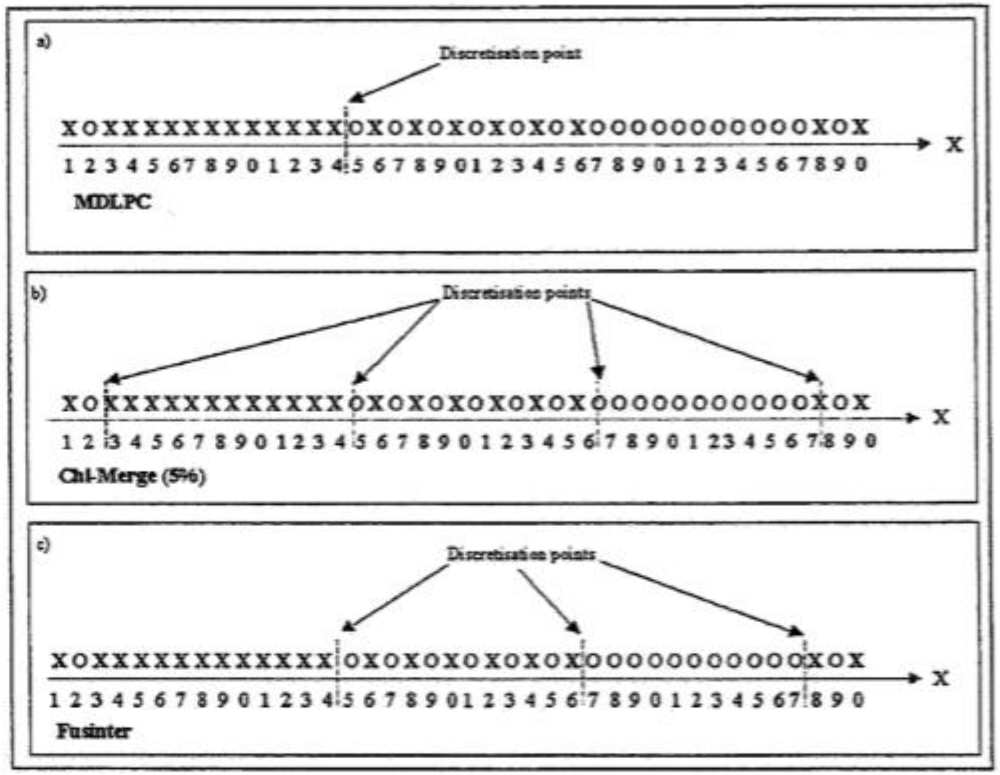
FUSINTER- Parameters

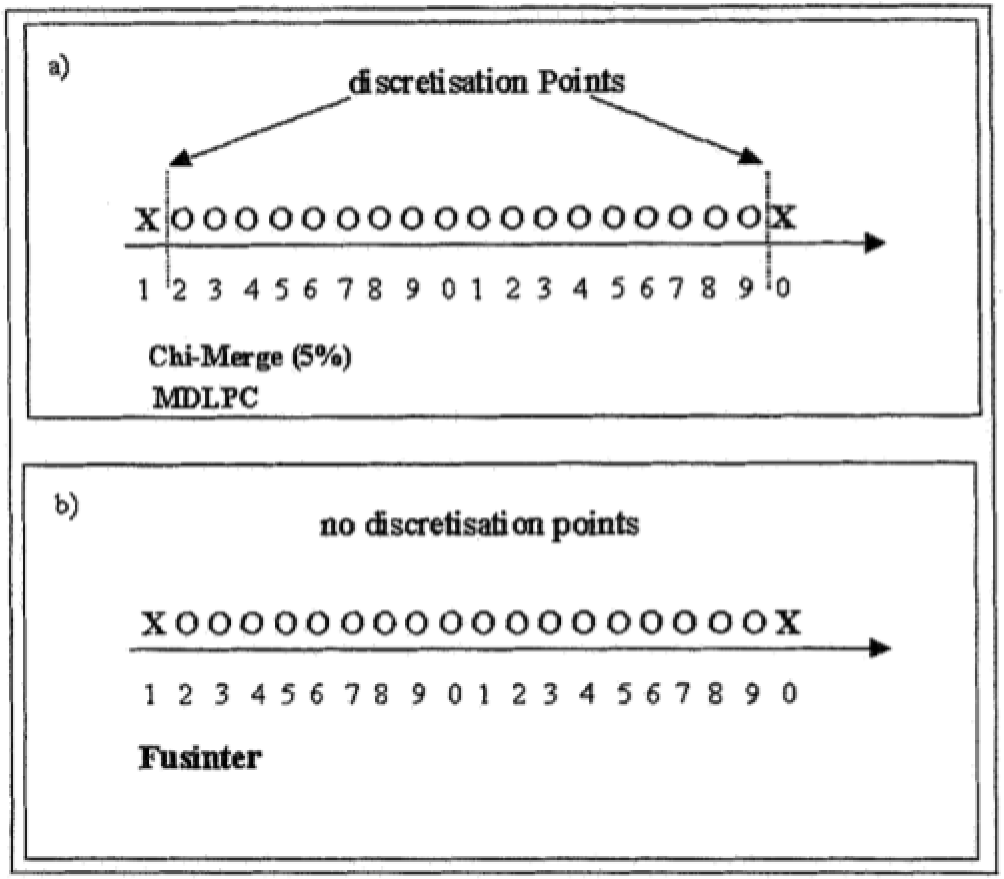
- Parameters $\alpha \ and \ \lambda$, controls the performance
- **How set them?**
  - By experiment: Cross validation
  - Theoretically: force the behavior of the method in particular situations
    1. **Minimize the number of intervals having a too small size**
    2. **Choose λ by maximizing uncertainty – prevent over-splitting**

FUSINTER- Cont.’






a)
Discretisation point
MDLPC
b)
Discretisation points
Chi-Merge (5%)
c)
Discretisation points
Fusinter


a)
discretisation Points
Chi-Merge (5%)
MDLPC
b)
no discretisation points
Fusinter

Online Discretization

- Standard discretization algorithms : needs entire dataset in main memory
- Industry outputs data in form of batches or individual instances(online)

Challenges

- Interval labeling
- Constantly revise their time and memory requirement
- Concept drift
    1. External drift detector
    2. Self-adaptive strategy : sliding window, ensembles, build model incrementally
- Relation with online learner (1-step definition)

Ideas

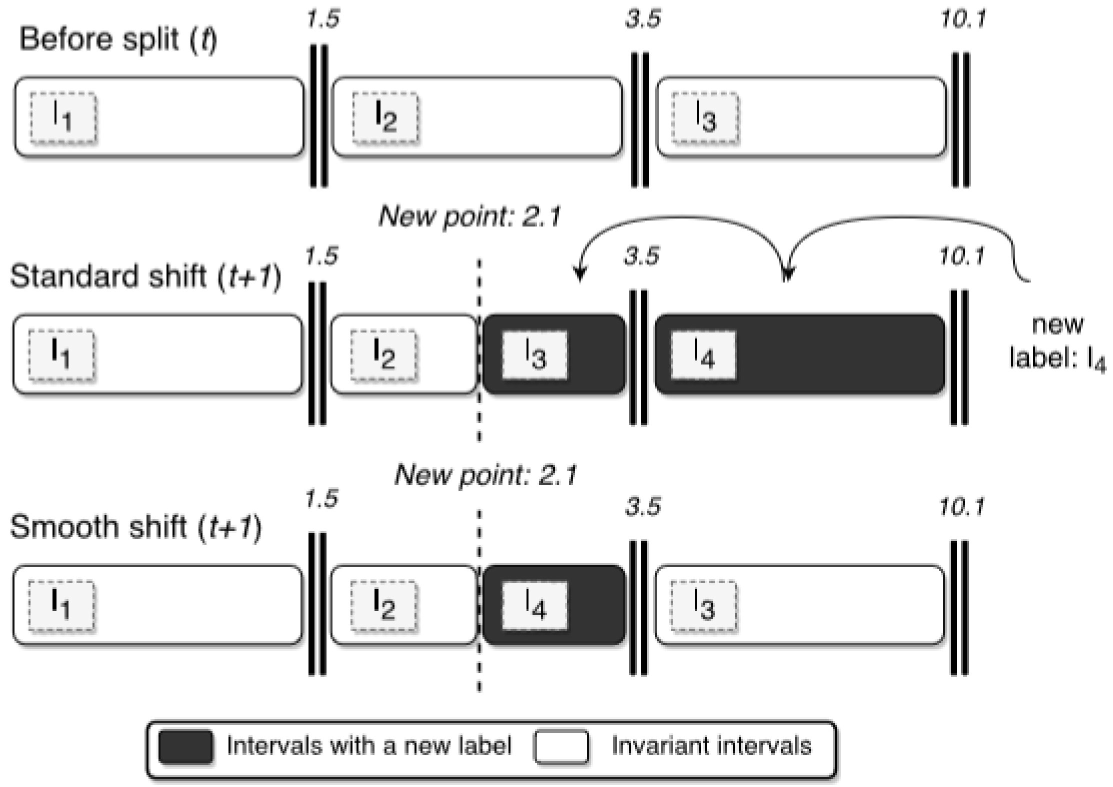
- Unsupervised : equal width, equal frequency (IDA), and etc.
- Supervised : PiD, LOFD, and etc.
- **PiD**:
  - Two layer approach:
    1. Produce preliminary cut-points by summarizing data
    2. merge
- **Problems**
  1. Correspondence between layers as time passes
  2. High skewness data
  3. Repetitive values

LOFD

- Local Online Fusion Discretizer
- Ramirez, Garcia, and Herrera (2018)
- Supervised
- Online
- Local
- Bottom-up
- Entropy Based
- Self-adaptive

LOFD – Interval Labeling

- Smooth Shifting



LOFD- Algorithm

- **High-informative splits:**
  - Hard to track distributions due to memory bound
  - More accurately distribution of intervals -> wiser decisions
  - Independent **memory constrained Histogram**
- **Bi-directional discretization:**
  - Both **splits** and **merging** (inserting and removing)
  - Although, merges are naturally applied, splits are more complex
- **Extended merges:**
  - Local changes -> adjacent merges
  - For splits, adjacent intervals are considered

LOFD- Algorithm - Cont.’

- **Split:**
- When the new value is a boundary point
- **Merge:**
- Resulting interval’s quadratic entropy is lower than sum of the parts
- **Data Structures:**
    - Red-black tree, Queue, and Histogram

LOFD- Algorithm - Cont.’

Algorithm 1 LOFD algorithm

INPUT: D, initTh, maxHist
// D is the input dataset.
// *initTh* Number of instances before initializing intervals
// *maxHist* Maximum number of elements in interval histograms
I = On the first batch ($i = 1 \dots initTh$), apply the static discretization process explained in [20].
for $i = initTh + 1 \to N$ **do**
  for $A \in M$ **do**
    $ceil$ = retrieve the ceiling interval that contains D_{iA}
    if $ceil \neq null$ **then**
      $isBound$ = check if D_{iA} is boundary
      Insert D_{iA} into $ceil$ and update its criterion
      if $isBound == true$ **then**
        $(ceil, new)$ = split $ceil$ into two intervals with D_{iA} as cutpoint
        Evaluate local merges between $ceil$, new, and the surrounding intervals until no improvement is achieved.
        Insert the resulting set into I_A
      end if
    else
      $last$ = Create a new interval on the right side with D_{iA} as upper limit
      Insert $last$ into I_A
      Evaluate merge with the old maximum interval
    end if
  end for
  add D_i to the timestamped queue
  for $int \in I_A$ **do**
    if $|histogram(int)| > maxHist$ **then**
      Remove old points from the timestamped queue, and subsequently, from the local histograms until $|histogram(int)| <= maxHist$. Remove empty intervals.
    end if
  end for
end for

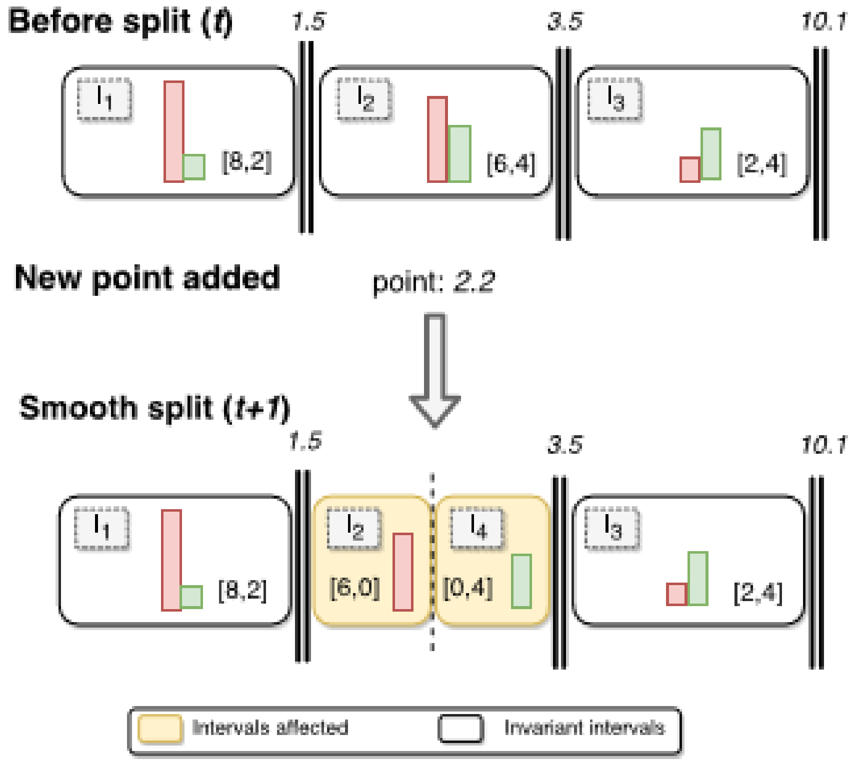


Table 8: Number of intervals generated by discretizer. *Best value (lowest) by row is highlighted in bold.*

	PiD	IDA	OC	LOFD
airlines	**17**	48	29	39
elecNormNew	81	54	**33**	50
kddcup_10	300	**138**	158	153
poker-lsn	51	55	43	**42**
covtypeNorm	344	330	96	**82**
blips	1,924	126	**120**	552
sudden_drift	22	24	**18**	28
gradual_drift_med	**17**	24	18	30
gradual_recurring_drift	1,829	126	**120**	504
incremental_fast	1,085	66	60	**55**
incremental_slow	313	66	**60**	75
MEAN	543.91	96.09	**68.64**	146.36

Table 6: Classification test accuracy on discretized data. Hoeffding tree used as learner.

	PiD	IDA	OC	HT	LOFD
airlines	64.3951	64.5158	**65.3619**	65.0784	65.0008
elecNormNew	79.8442	79.8354	70.2132	79.1954	**80.7645**
kddcup_10	**99.8389**	99.7929	99.8368	99.7413	99.5120
poker-lsn	57.9820	69.8381	55.4892	76.0685	**76.1936**
covtypeNorm	77.6671	75.8652	70.1681	80.3119	**81.8190**
blips	73.6652	86.0112	35.7974	**90.9808**	79.3036
sudden_drift	69.5128	82.9856	61.3936	84.8418	**86.7238**
gradual_drift_med	64.6858	84.1394	51.1838	85.5088	**86.5246**
gradual_recurring_drift	68.2206	83.7164	35.6192	**88.3368**	77.8664
incremental_fast	71.1508	78.6526	50.6528	**82.7748**	77.0852
incremental_slow	66.3744	76.7644	50.5308	**83.1052**	70.9906
MEAN	72.1215	80.1924	58.7497	**83.2676**	80.1622

References

[1] D. A. Zighed, S. Rabas´eda, R. Rakotomalala, FUSINTER: A method660 for discretization of continuous attributes, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (3) (1998) 307–326.

[2] S. Ramirez-Gallegoa, S. Garciaa, F. Herreraa,b Online Entropy-Based Discretization for Data Streaming Classification, 2018

[3] Huan Liu, Farhad Hussain, Chew lim Tan, Manoranjan Dash Discretization: An Enabling Technique, (2002), 393-423

THANK YOU :D