# Logarithmic Time Online Multiclass prediction

**Anna Choromanska**

Courant Institute of Mathematical Sciences
New York University

# eXtreme multi-class classification problem

**Problem setting**:

- classification with large number of classes
- data is accessed online

**Goal**:

- good predictor with **logarithmic training and testing time**
- reduction to tree-structured binary classification
- top-down approach for class partitioning allowing gradient descent style optimization

# eXtreme multi-class classification problem











# eXtreme multi-class classification problem

# eXtreme multi-class classification problem

# eXtreme multi-class classification problem

# eXtreme multi-class classification problem

# eXtreme multi-class classification problem

# eXtreme multi-class classification problem

$\cdots$

# What was already done...

- Intractable
  - one-against-all [RK04]
  - variants of ECOC [DB95], e.g. PECOC [LB05]
  - clustering-based approaches [BWG10, WMY13]
- Choice of partition not addressed
  - Filter Tree and error-correcting tournaments [BLR09]
- Choice of partition addressed, but dedicated to conditional probability estimation
  - conditional probability tree [BLLSS09]
- Splitting criteria not well-suited to large class setting
  - decision trees [KM95]
- ...

# $\mathcal{O}(\log k)$ time

**Theorem**

*There exists multi-class classification problems where achieving* $0$ *error rate requires* $\Omega(\log k)$ *time to train or test per example.*

**Proof.**

Follows from information theory[CT91].

Any multi-class classification algorithm must uniquely specify the bits of all labels that it predicts correctly on. Consequently, Kraft's inequality [CT91, Equation 5.6] implies that the expected *computational* complexity of predicting correctly is $\Omega(H(Y))$ per example where $H(Y)$ is the Shannon entropy of the label. For the worst case distribution on $k$ classes, this implies $\Omega(\log(k))$ computation is required. $\square$

# How do you learn the structure?

- Not all partitions are equally difficult, e.g.
  if you do $\{1, 7\}$ vs $\{3, 8\}$, the next problem is hard;
  if you do $\{1, 8\}$ vs $\{3, 7\}$, the next problem is easy;
  if you do $\{1, 3\}$ vs $\{7, 8\}$, the next problem is easy.

# How do you learn the structure?

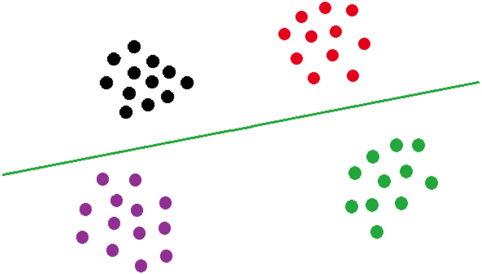
- Not all partitions are equally difficult, e.g.
  if you do $\{1,7\}$ vs $\{3,8\}$, the next problem is hard;
  if you do $\{1,8\}$ vs $\{3,7\}$, the next problem is easy;
  if you do $\{1,3\}$ vs $\{7,8\}$, the next problem is easy.
- [BWG10]: Better to confuse near leaves than near root.
  Intuition: The root predictor tends to be overconstrained while the leafwards predictors are less constrained.

# How do you learn the structure?

**Our approach** [**C**L15, **C**CB15, **C**AL13]:

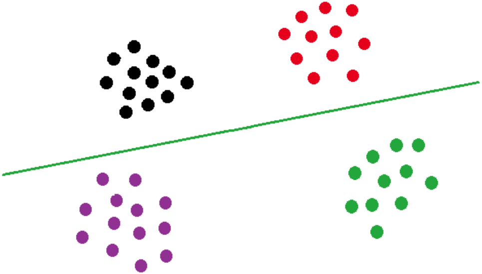
- **top-down** approach for class partitioning
- splitting criterion guaranteeing
  **balanced tree** $\Rightarrow$ **logarithmic training and testing time**
  <u>and</u>
  **small classification error**

# Pure split and balanced split



- $k_r(x)$: number of data points in the same class as $x$ on the right side of the partitioning
- $k(x)$: total number of data points in the same class as $x$
- $n_r$: number of data points on the right side of the partitioning
- $n$: total number of data points

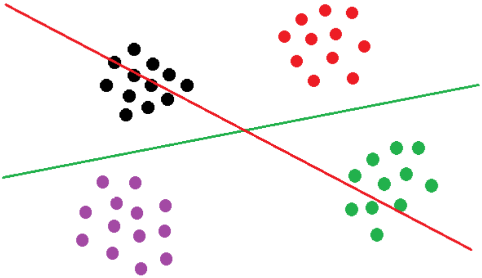
# Pure split and balanced split



- $k_r(x)$: number of data points in the same class as $x$ on the right side of the partitioning
- $k(x)$: total number of data points in the same class as $x$
- $n_r$: number of data points on the right side of the partitioning
- $n$: total number of data points

**Measure of balancedness:** $\frac{\mathbf{n_r}}{\mathbf{n}}$

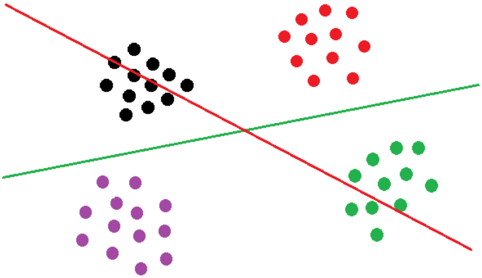
# Pure split and balanced split



- $k_r(x)$: number of data points in the same class as $x$ on the right side of the partitioning
- $k(x)$: total number of data points in the same class as $x$
- $n_r$: number of data points on the right side of the partitioning
- $n$: total number of data points

**Measure of balancedness:** $\mathbf{\frac{n_r}{n}}$

# Pure split and balanced split



- $k_r(x)$: number of data points in the same class as $x$ on the right side of the partitioning
- $k(x)$: total number of data points in the same class as $x$
- $n_r$: number of data points on the right side of the partitioning
- $n$: total number of data points

**Measure of balancedness:** $\frac{n_r}{n}$ **Measure of purity:** $\frac{k_r(x)}{k(x)}$

# Pure split and balanced split

- $k$: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

# Pure split and balanced split

- $k$: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

**Definition (Balanced split)**

The hypothesis $h \in \mathcal{H}$ induces a balanced split iff

$$\exists_{c \in (0,0.5]} c \leq \text{balance} \leq 1 - c.$$

# Pure split and balanced split

- $k$: number of classes
- $\mathcal{H}$: hypothesis class (typically: linear classifiers)
- $\pi_y = \frac{|\mathcal{X}_y|}{n}$
- balance $= Pr(h(x) > 0)$
- purity $= \sum_{y=1}^{k} \pi_y \min(Pr(h(x) > 0|y), Pr(h(x) < 0|y))$

**Definition (Balanced split)**

The hypothesis $h \in \mathcal{H}$ induces a balanced split iff

$$\exists_{c \in (0,0.5]} c \leq \text{balance} \leq 1 - c.$$

**Definition (Pure split)**

The hypothesis $h \in \mathcal{H}$ induces a pure split iff

$$\exists_{\delta \in [0,0.5)} \text{purity} \leq \delta.$$

# Objective function

$$J(h) = 2\sum_{y=1}^{k} \pi_y \left| P(h(x) > 0) - P(h(x) > 0|y) \right|$$

$$= 2\mathbb{E}_{x,y}\left[ \left| P(h(x) > 0) - P(h(x) > 0|y) \right| \right]$$

**J(h) $\Rightarrow$ Splitting criterion (objective function)**

Given a set of $n$ examples each with one of $k$ labels, find a **partitioner** $h$ that maximizes the objective.

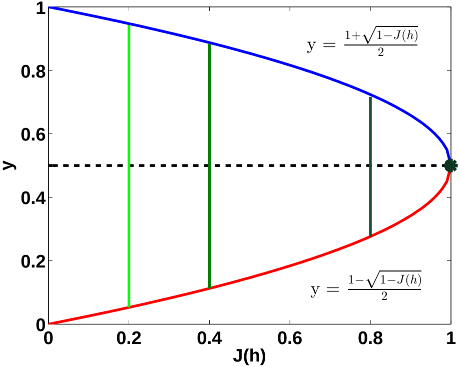
**Lemma**

*For any hypothesis* $h : \mathcal{X} \mapsto \{-1, 1\}$, *the objective* $J(h)$ *satisfies* $J(h) \in [0, 1]$. *Furthermore,* $h$ *induces a maximally pure and balanced partition iff* $J(h) = 1$.

# Balancing and purity factors

- **Balancing factor**

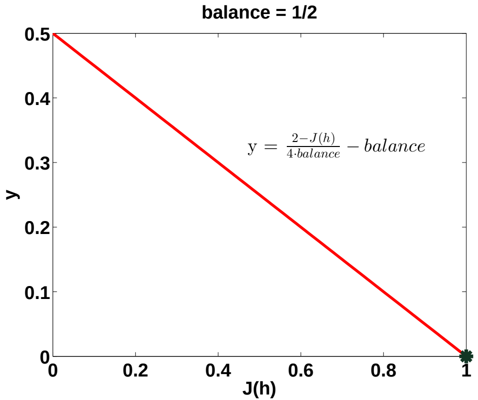
$$\text{balance} \in \left[\frac{1-\sqrt{1-J(h)}}{2}, \frac{1+\sqrt{1-J(h)}}{2}\right]$$



# Balancing and purity factors

- **Purity factor**

$$\text{purity} \leq \frac{2 - J(h)}{4 \cdot \text{balance}} - \text{balance}$$



# What is the quality of obtained tree?

- In each node of the tree $\mathcal{T}$ optimize the splitting criterion
- Apply recursively to construct a tree structure
- Measure the quality of the tree using entropy

$$G_{\mathcal{T}} = \sum_{l \in \text{leafs of } \mathcal{T}} w_l \sum_{y=1}^{k} \pi_{l,y} \ln\left(\frac{1}{\pi_{l,y}}\right)$$

Why?

Small entropy of leafs $\Rightarrow$ pure leafs

Goal: maximizing the objective reduces the entropy

# What is the quality of obtained tree?

## Definition (Weak Hypothesis Assumption)

Let $m$ denotes any node of the tree $\mathcal{T}$, and let $\beta_m = P(h_m(x) > 0)$ and $P_{m,y} = P(h_m(x) > 0|y)$. Furthermore, let $\gamma \in \mathbb{R}^+$ be such that for all $m$, $\gamma \in (0, \min(\beta_m, 1 - \beta_m)]$. We say that the *weak hypothesis assumption* is satisfied when for any distribution $\mathcal{P}$ over $\mathcal{X}$ at each node $m$ of the tree $\mathcal{T}$ there exists a hypothesis $h_m \in \mathcal{H}$ such that $J(h_m)/2 = \sum_{y=1}^{k} \pi_{m,y}|P_{m,y} - \beta_m| \geq \gamma$.

## Theorem

*Under the Weak Hypothesis Assumption, for any $\epsilon \in [0, 1]$, to obtain $G_{\mathcal{T}} \leq \epsilon$ it suffices to make $\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}$ splits.*

# What is the quality of obtained tree?

## Definition (Weak Hypothesis Assumption)

Let $m$ denotes any node of the tree $\mathcal{T}$, and let $\beta_m = P(h_m(x) > 0)$ and $P_{m,y} = P(h_m(x) > 0|y)$. Furthermore, let $\gamma \in \mathbb{R}^+$ be such that for all $m$, $\gamma \in (0, \min(\beta_m, 1 - \beta_m)]$. We say that the *weak hypothesis assumption* is satisfied when for any distribution $\mathcal{P}$ over $\mathcal{X}$ at each node $m$ of the tree $\mathcal{T}$ there exists a hypothesis $h_m \in \mathcal{H}$ such that $J(h_m)/2 = \sum_{y=1}^{k} \pi_{m,y}|P_{m,y} - \beta_m| \geq \gamma$.

## Theorem

*Under the Weak Hypothesis Assumption, for any $\epsilon \in [0, 1]$, to obtain $G_{\mathcal{T}} \leq \epsilon$ it suffices to make $\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}$ splits.*

- Tree depth $\approx \log\left[\left(\frac{1}{\epsilon}\right)^{\frac{4(1-\gamma)^2 \ln k}{\gamma^2}}\right] = \mathcal{O}(\ln k) \Rightarrow$

$\Rightarrow$ **logarithmic training and testing time**

# What is the quality of obtained tree?

Connection to other entropy functions, like

Gini-entropy:
$G_{\mathcal{T}} = \sum_{l \in \text{leafs of } \mathcal{T}} w_l \sum_{y=1}^{k} \pi_{l,y}(1 - \pi_{l,y})$
and its modified version:
$G_{\mathcal{T}} = \sum_{l \in \text{leafs of } \mathcal{T}} w_l \sum_{y=1}^{k} \sqrt{\pi_{l,y}(\mathcal{C} - \pi_{l,y})}$

can be found in [**C**CB15].

# LOMtree algorithm

- Recall the objective function we consider at every tree node

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

Problem: discrete optimization
Relaxation: drop the indicator operator and look at margins

# LOMtree algorithm

- Recall the objective function we consider at every tree node

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

Problem: discrete optimization
Relaxation: drop the indicator operator and look at margins

# LOMtree algorithm

- Recall the objective function we consider at every tree node

$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[\mathbb{1}(h(x) > 0)] - \mathbb{E}_x[\mathbb{1}(h(x) > 0|y)]|].$$

Problem: discrete optimization
Relaxation: drop the indicator operator and look at margins

- The objective function becomes

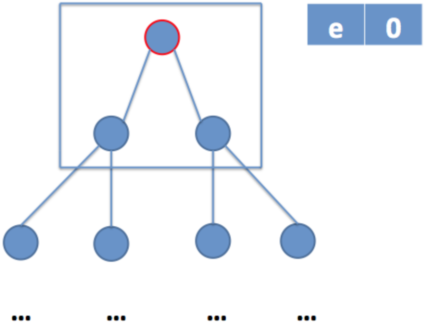
$$J(h) = 2\mathbb{E}_y[|\mathbb{E}_x[h(x)] - \mathbb{E}_x[h(x)|y]|].$$

  - Keep the online empirical estimates of these expectations.
  - The sign of the difference of two expectations decides whether to send an example to the left or right child node.

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

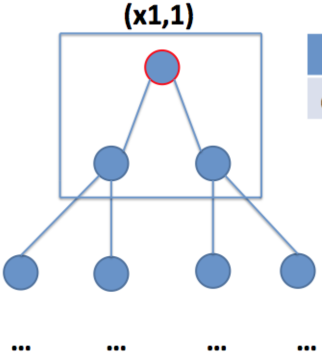
- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

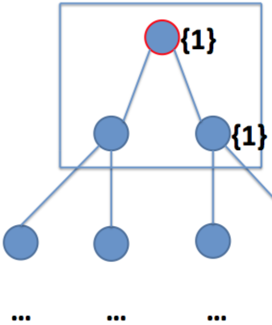



| e | 0 |
| --- | --- |
| e1 | 0 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

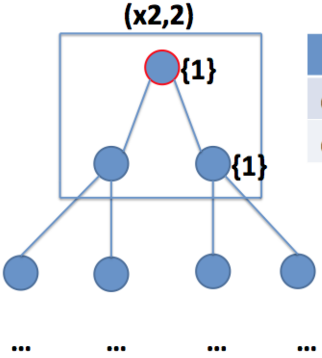



| e | 1 |
|---|---|
| e1 | 1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



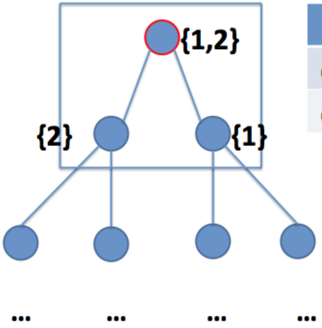

| e | 1 |
|---|---|
| e1 | 1 |
| e2 | 0 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$

For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

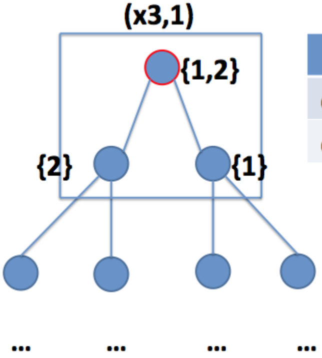



| e | 0 |
|---|---|
| e1 | 1 |
| e2 | -1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



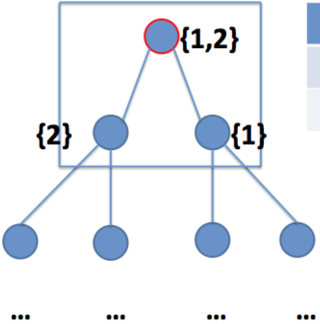

| e | 0 |
|---|---|
| e1 | 1 |
| e2 | -1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$

For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



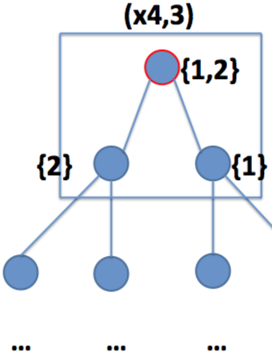

| e | 1/3 |
|---|---|
| e1 | 1 |
| e2 | -1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$

For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

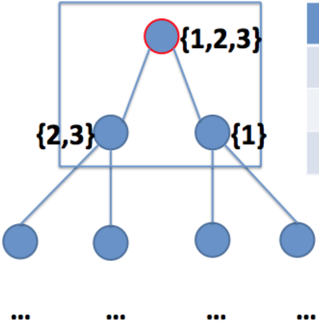



| e | 1/3 |
|---|---|
| e1 | 1 |
| e2 | -1 |
| e3 | 0 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

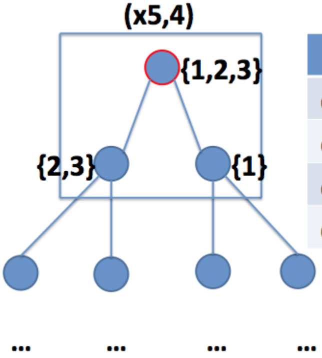



| e | 0 |
|---|---|
| e1 | 1 |
| e2 | -1 |
| e3 | -1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

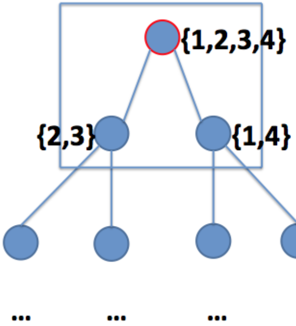



| e | 0 |
|---|---|
| e1 | 1 |
| e2 | -1 |
| e3 | -1 |
| e4 | 0 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$



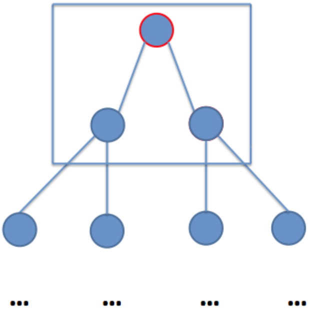

| e | 3/5 |
|---|---|
| e1 | 1 |
| e2 | -1 |
| e3 | -1 |
| e4 | 1 |

# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$
For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to construct a tree structure.



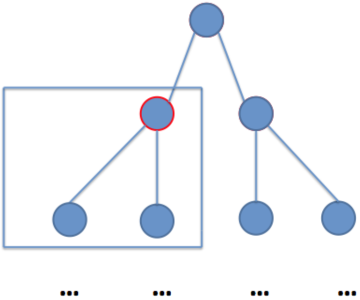
# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$

For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to construct a tree structure.



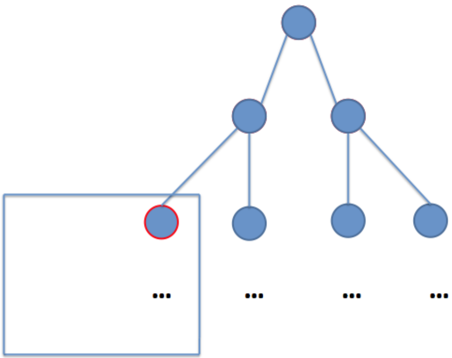
# LOMtree algorithm

Let $e = 0$ and for all $y$, $e_y = 0$, $n_y = 0$

For each example $(x, y)$

- if $e_y < e$ then $b = -1$ else $b = 1$
- Update $w$ using $(x, b)$
- $n_y \leftarrow n_y + 1$
- $e_y \leftarrow \frac{(n_y - 1)e_y}{n_y} + \frac{w.x}{n_y}$
- $e \leftarrow \frac{(n-1)e}{n} + \frac{w.x}{n}$

Apply recursively to construct a tree structure.



# Pseudo-code

**Input**: regression algorithm $R$, max number of tree non-leaf nodes $T$, swap resistance $R_S$

Subroutine **SetNode** ($v$)

$\mathbf{m}_v = \emptyset$ ($\mathbf{m}_v(y)$ - sum of the scores for class $y$)
$\mathbf{l}_v = \emptyset$ ($\mathbf{l}_v(y)$ - number of points of class $y$ reaching $v$)
$\mathbf{n}_v = \emptyset$ ($\mathbf{n}_v(y)$ - number of points of class $y$ which are used to train regressor in $v$)
$\mathbf{e}_v = \emptyset$ ($\mathbf{e}_v(y)$ - expected score for class $y$)
$\mathbf{E}_v = 0$ (expected total score)
$C_v = 0$ (the size of the smallest leaf in the subtree with root $v$)

Subroutine **UpdateC** ($v$)

**While** ($v \neq r$ AND $C_{\text{PARENT}(v)} \neq C_v$) $\{v = \text{PARENT}(v);\quad C_v = \min(C_{\text{LEFT}(v)}, C_{\text{RIGHT}(v)})\}$

**Create** root $r = 0$: **SetNode** ($r$); $t = 1$
**For each** example $(\mathbf{x}, y)$ **do**
  Set $j = r$
  **While** $j$ is not a leaf **do**
    **If** ($l_j(y) = \emptyset$)
      $m_j(y) = 0$; $l_j(y) = 0$; $n_j(y) = 0$; $e_j(y) = 0$
    **If** ($E_j > \mathbf{e}_j(y)$) $c = -1$ **Else** $c = 1$
    **Train** $h_j$ with example $(\mathbf{x}, c)$: $R(\mathbf{x}, c)$
    $\mathbf{l}_j(y){+}{+}$; $\mathbf{n}_j(y){+}{+}$; $\mathbf{m}_j(y) \mathrel{+}= h_j(\mathbf{x})$; $\mathbf{e}_j(y) = \mathbf{m}_j(y)/\mathbf{n}_j(y)$; $E_j = \frac{\sum_{i=1}^{k} \mathbf{m}_j(i)}{\sum_{i=1}^{k} \mathbf{n}_j(i)}$]
    **Set** $j$ to the child of $j$ corresponding to $h_j$
    **If**($j$ is a leaf)
      $\mathbf{l}_j(y){+}{+}$
      **If**($\mathbf{l}_j$ has at least 2 non-zero entries)
        **If**($t<T$ OR $C_j - \max_i \mathbf{l}_j(i) > R_S(C_r+1)$)
          **If** ($t<T$)
            **SetNode** ($\text{LEFT}(j)$); **SetNode** ($\text{RIGHT}(j)$); $t{+}{+}$
          **Else** **Swap**(j)
          $C_{\text{LEFT}(j)} = \lfloor * C_j/2$; $C_{\text{RIGHT}(j)} = C_j - C_{\text{LEFT}(j)}$; **UpdateC** ($\text{LEFT}(j)$)
    $C_j{+}{+}$

# Swapping

Subroutine **Swap** (v)

---

Find a leaf $s$ for which $(C_s = C_r)$
$s_{\text{PA}}=\text{PARENT}(s)$; $s_{\text{GPA}}=\text{GRANDPA}(s)$; $s_{\text{SIB}}=\text{SIBLING}(s)$
**If** $(s_{\text{PA}} = \text{LEFT}(s_{\text{GPA}}))$ $\text{LEFT}(s_{\text{GPA}}) = s_{\text{SIB}}$ **Else** $\text{RIGHT}(s_{\text{GPA}}) = s_{\text{SIB}}$
**UpdateC** $(s_{\text{SIB}})$; **SetNode** $(s)$; $\text{LEFT}(v) = s$; **SetNode** $(s_{\text{PA}})$; $\text{RIGHT}(v) = s_{\text{PA}}$

---

Node $j$ splits if the following holds:

$$C_j - \max_{i \in \{1,2,\dots,k\}} \mathbf{I}_j(i) > R_S(C_r + 1),$$

# Swapping

Subroutine **Swap** (v)

---

Find a leaf $s$ for which $(C_s = C_r)$

$s_{\text{PA}}{=}\text{PARENT}(s)$; $s_{\text{GPA}} = \text{GRANDPA}(\mathrm{s})$; $s_{\text{SIB}}{=}\text{SIBLING}(s)$

**If** $(s_{\text{PA}} = \text{LEFT}(s_{\text{GPA}}))$ $\text{LEFT}(s_{\text{GPA}}) = s_{\text{SIB}}$ **Else** $\text{RIGHT}(s_{\text{GPA}}) = s_{\text{SIB}}$

**UpdateC** $(s_{\text{SIB}})$; **SetNode** $(s)$; $\text{LEFT}(v) = s$; **SetNode** $(s_{\text{PA}})$; $\text{RIGHT}(v) = s_{\text{PA}}$

---

Node $j$ splits if the following holds:

$$C_j - \max_{i \in \{1,2,\dots,k\}} \mathbf{I}_j(i) > R_S(C_r + 1),$$

**Lemma**

*Let the swap resistance $R_S$ be greater or equal to 4. Then for all sequences of examples, the number of times Algorithm* **??** *recycles any given node is upper-bounded by the logarithm (with base 2) of the sequence length.*

# Swapping

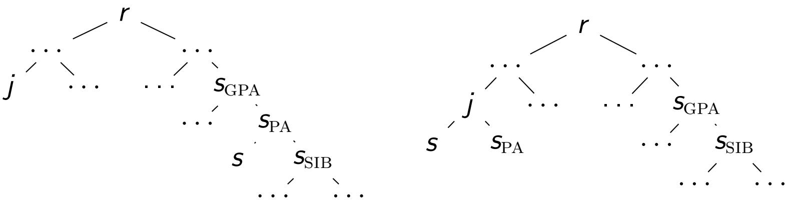



Figure : Illustration of the swapping procedure. **Left:** before the swap, **right:** after the swap.

# Experiments

Table : Training time on selected problems.

| | Isolet | Sector | Aloi |
|---|---|---|---|
| LOMtree | **16.27s** | **12.77s** | **51.86s** |
| OAA | 19.58s | 18.37s | 11m2.43s |

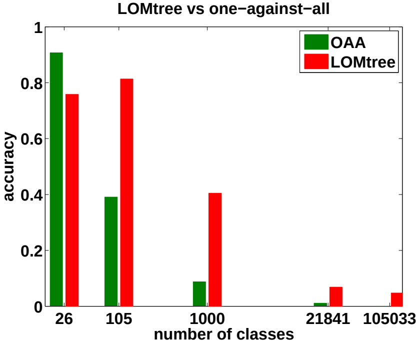
Table : Per-example test time on all problems.

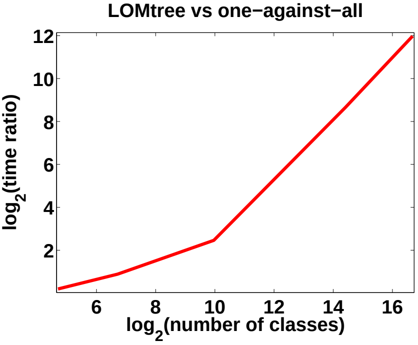
| | Isolet | Sector | Aloi | ImNet | ODP |
|---|---|---|---|---|---|
| LOMtree | **0.14ms** | **0.13ms** | **0.06ms** | **0.52ms** | **0.26ms** |
| OAA | 0.16 ms | 0.24ms | 0.33ms | 0.21s | 1.05s |

Table : Test error (%) and confidence interval on all problems.

| | LOMtree | Rtree | Filter tree |
|---|---|---|---|
| Isolet (26) | **6.36**$\pm$1.71 | 16.92$\pm$2.63 | 15.10$\pm$2.51 |
| Sector (105) | 16.19$\pm$2.33 | **15.77**$\pm$2.30 | 17.70$\pm$2.41 |
| Aloi (1000) | **16.50**$\pm$0.70 | 83.74$\pm$0.70 | 80.50$\pm$0.75 |
| ImNet (22K) | **90.17**$\pm$0.05 | 96.99$\pm$0.03 | 92.12$\pm$0.04 |
| ODP (105K) | **93.46**$\pm$0.12 | 93.85$\pm$0.12 | 93.76$\pm$0.12 |

# Experiments



# Conclusions

**New algorithm:**

- **Reduction** from multi-class to binary classification
- New **splitting criterion** with desirable properties
- **Logarithmic training and testing time**

# References

Logarithmic Time Online Multiclass prediction, Anna Choromanska, John Langford, NIPS 2015

On the boosting ability of top-down decision tree learning algorithm for multiclass classiffication, A. Choromanska[1], K. Choromanski[1], M. Bojarski, 2015 (submitted)

[1]equal contribution

# Acknowledgments

We would like to thank Alekh Agarwal, Dean Foster, Robert Schapire and Matus Telgarsky for valuable discussions.

# Thank you!!!