Non-Linear Gradient Boosting for Class-Imbalance Learning

By

Jordan Fréry, Amaury Habrard, Marc Sebban and Liyun He-Guelton
Class Imbalance

Many real life applications suffer from the class imbalance problem:
Class Imbalance

Many real life applications suffer from the class imbalance problem:

• Anomaly detection
Class Imbalance

Many real life applications suffer from the class imbalance problem:

- Anomaly detection
- Spam detection
Class Imbalance

Many real life applications suffer from the class imbalance problem:

- Anomaly detection
- Spam detection
- Click prediction
Class Imbalance

Many real life applications suffer from the class imbalance problem:

• Anomaly detection
• Spam detection
• Click prediction
• Fraud detection
Class Imbalance

Many real life applications suffer from the class imbalance problem:

- Anomaly detection
- Spam detection
- Click prediction
- Fraud detection
- ...

[Graphs showing data distribution with a significant imbalance]
Class Imbalance

Many real life applications suffer from the class imbalance problem:

- Anomaly detection
- Spam detection
- Click prediction
- Fraud detection
- ...

Atos Worldline focuses on fraud detection over credit card transactions. Only 0.2% of the examples are fraudulent transactions.
Supervised imbalanced learning

• Goal is to find a function $f : X \rightarrow Y$
Supervised imbalanced learning

- Goal is to find a function $f : X \rightarrow Y$

- We assume that there is an unknown joint distribution $D$ over $X \times Y$
Supervised imbalanced learning

• Goal is to find a function $f : X \rightarrow Y$

• We assume that there is an unknown joint distribution $D$ over $X \times Y$

• We have a training set of $M$ examples $\{x_i, y_i\}_{n=1}^{M} \in (X \times Y)^M$ i.i.d. according to $D$
Supervised imbalanced learning

• Goal is to find a function $f : X \rightarrow Y$

• We assume that there is an unknown joint distribution $D$ over $X \times Y$

• We have a training set of $M$ examples $\{x_i, y_i\}_{i=1}^M \in (X \times Y)^M$ i.i.d. according to $D$

• We have $P$ and $N$, the number of examples from the positive and negative class respectively with $N >> P$. 
Supervised imbalanced learning

• The straightforward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  - Undersampling: loss of information
  - Oversampling: risk of overfitting
  - Cost sensitive learning: hard to find the right cost
  - Need to adjust the decision threshold for the real class distribution

• We rather deal with the original dataset.
Supervised imbalanced learning

• The straight forward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  – Undersampling: loss of information
Supervised imbalanced learning

• The straight forward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  – Undersampling: loss of information
  – Oversampling: risk of overfitting
Supervised imbalanced learning

• The straight forward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  – Undersampling: loss of information
  – Oversampling: risk of overfitting
  – Cost sensitive learning: hard to find the right cost

• We rather deal with the original dataset.
Supervised imbalanced learning

• The straightforward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  – Undersampling: loss of information
  – Oversampling: risk of overfitting
  – Cost sensitive learning: hard to find the right cost
  – Need to adjust the decision threshold for the real class distribution
Supervised imbalanced learning

• The straightforward approach is to use sampling or cost sensitive based methods. However they have some drawbacks:
  – Undersampling: loss of information
  – Oversampling: risk of overfitting
  – Cost sensitive learning: hard to find the right cost
  – Need to adjust the decision threshold for the real class distribution

• We rather deal with the original dataset.
Boosting combines different hypotheses, $h_1, \ldots, h_T$, linearly with their respective weight $\alpha_1, \ldots, \alpha_T$
Boosting

Boosting combines different hypotheses, $h_1, ..., h_T$, linearly with their respective weight $\alpha_1, ..., \alpha_T$:

$$F_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$
Boosting

Boosting combines different hypotheses, $h_1, \ldots, h_T$, linearly with their respective weight $\alpha_1, \ldots, \alpha_T$:

$$F_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Adaboost, Logitboost, Rankboost, Gentleboost, Brownboost, Lpboost, Gradient boosting, ... They all follow the same schema.
• Automatically reweights the example

• Focus on the minority class but...

why does it take so many iterations?

Few iterations >100 iterations
• Automatically reweights the example
- Automatically reweights the example
- Focus on the minority class but..

![Few iterations](image1.png)

![>100 iterations](image2.png)
• Automatically reweights the example
• Focus on the minority class but..

why does it take so many iterations?
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

- • Learn stronger weak learner? (counterintuitive, high risk of overfitting)
- • Use more complex combinations of the weak learners (non-linear)
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

- Learn stronger weak learner? (counterintuitive, high risk of overfitting)
- Use more complex combinations of the weak learners (non-linear)
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

• Learn stronger weak learner (counterintuitive, high risk of overfitting)
• Use more complex combinations of the weak learners (non-linear)
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

Solutions?
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

Solutions?

• Learn stronger weak learner? (counterintuitive, high risk of overfitting)
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

Solutions?
- Learn stronger weak learner? (counterintuitive, high risk of overfitting)
- Use more complex combinations of the weak learners (non-linear)
Gradient Boosting

At iteration $t - 1$ we find the residuals for all $\{x_i\}_{i=1}^{M}$

$$r_t(x_i) = - \frac{\partial L(y_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}$$
Gradient Boosting

At iteration $t - 1$ we find the residuals for all $\{x_i\}_{i=1}^M$

$$r_t(x_i) = -\frac{\partial L(y_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}$$

We find a model $h_t$ with its corresponding weight such that:

$$h_t = \arg\min_h \sum_{i=1}^M (h(x_i) - r(x_i))^2$$
Gradient Boosting

At iteration $t - 1$ we find the residuals for all $\{x_i\}_{i=1}^M$

$$r_t(x_i) = -\frac{\partial L(y_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}$$

We find a model $h_t$ with its corresponding weight such that:

$$h_t = \arg\min_h \sum_{i=1}^M (h(x_i) - r(x_i))^2$$

$$\alpha_t = \arg\min_\alpha \sum_{i=1}^M L(y_i, f_{t-1}(x_i) + \alpha h_t(x_i))$$
Non-Linear Gradient Boosting

Idea:

- Build different representations of the weak learners.
Non-Linear Gradient Boosting

Idea:

- Build different representations of the weak learners.
- Combine these representations non-linearly.
Non-Linear Gradient Boosting

Idea:

• Build different representations of the weak learners.
• Combine these representations non-linearly.
• Find a new $h$ that corrects the error of the non-linear combinations
Non-Linear Gradient Boosting

We define our new model $F$ as:
Non-Linear Gradient Boosting

We define our new model $F$ as:

$$F(x) = \sum_{r=1}^{R} \alpha^r L_r \left( \sum_{t=1}^{T} \alpha_t^r h_t(x) \right),$$

where $R$ is the number of combinations, $\alpha^r$ and $L_r$ the weight and the non-linear transformation of the combination $r$. 
Non-Linear Gradient Boosting

We define our new model $F$ as:

$$F(x) = \sum_{r=1}^{R} \alpha^r L_r \left( \sum_{t=1}^{T} \alpha^r_t h_t(x) \right),$$

where $R$ is the number of combinations, $\alpha^r$ and $L_r$ the weight and the non-linear transformation of the combination $r$. 
Non-Linear Gradient Boosting

We define our new model $F$ as:

$$F(x) = \sum_{r=1}^{R} \alpha^r \mathcal{L}_r \left( \sum_{t=1}^{T} \alpha_t^r h_t(x) \right),$$

where $R$ is the number of combinations, $\alpha^r$ and $\mathcal{L}_r$ the weight and the non-linear transformation of the combination $r$. 
Non-Linear Gradient Boosting

Boosted models

Non-linear combination

$F(x)$
In-Depth Analysis of NLB

Do the different representations bring divers information?

\[ C_{nm} = c_{vn} \times c_{vm}, \]

where \( n \) and \( m \) are the \( n \)-th and \( m \)-th representation.
In-Depth Analysis of NLB

Do the different representations bring divers information?

Correlation between the different combinations:

\[ C_{nm} = \frac{\text{cov}_{nm}}{\sqrt{\text{cov}_{nn} \times \text{cov}_{mm}}}, \]

where \( n \) and \( m \) are the \( n^{th} \) and \( m^{th} \) representation.
Are they all useful in the final model?
In-Depth Analysis of NLB

Are they all useful in the final model?

Importance of a combination:

\[ \Omega_r = \frac{1}{M} \sum_{i=1}^{M} \left( \alpha^r \mathcal{L}_r \left( \sum_{t=1}^{T} \alpha_t^r h_t(x_i) \right) \right) \]
In-Depth Analysis of NLB
In-Depth Analysis of NLB
Experiments

- 24 datasets from Keel repository
Experiments

- 24 datasets from Keel repository
- Imbalance ratio from 0.09 to 0.01
Experiments

• 24 datasets from Keel repository
• Imbalance ratio from 0.09 to 0.01

• Evaluation metrics:
  – $F_1$ score
  – Average precision
Experiments

• 24 datasets from Keel repository
• Imbalance ratio from 0.09 to 0.01

• Evaluation metrics:
  – $F_1$ score
  – Average precision

• Models for comparison
  – GB
  – NLB
Experiments

• 24 datasets from Keel repository
• Imbalance ratio from 0.09 to 0.01

• Evaluation metrics:
  – $F_1$ score
  – Average precision

• Models for comparison
  – GB
  – NLB

• We use 3-fold cross validation repeated 30 times.
## Experiments

In summary, over 24 datasets, NLB wins 20 times in terms of AP and 19 times in terms of F-score.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NLB(AP)</th>
<th>GB(AP)</th>
<th>NLB(F1)</th>
<th>GB(F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>poker-8 vs 6</td>
<td>29.3±19.8</td>
<td>25.8±31.3</td>
<td>28.9±24.4</td>
<td>9.8±19.8</td>
</tr>
<tr>
<td>abalone-20 vs 8-9-10</td>
<td>27.9±11.7</td>
<td>20.1±18.9</td>
<td>20.2±15.7</td>
<td>19.3±20.0</td>
</tr>
<tr>
<td>winequality-red-3 vs 5</td>
<td>8.7±6.0</td>
<td>11.1±12.3</td>
<td>7.2±14.0</td>
<td>2.8±7.9</td>
</tr>
<tr>
<td>winequality-white-3-9 vs 5</td>
<td>23.8±12.6</td>
<td>14.8±12.9</td>
<td>25.8±16.9</td>
<td>14.9±16.3</td>
</tr>
<tr>
<td>kr-vs-k-zero vs eight</td>
<td>99.0±1.5</td>
<td>95.2±7.0</td>
<td>77.1±7.3</td>
<td>81.5±16.4</td>
</tr>
<tr>
<td>winequality-red-8 vs 6-7</td>
<td>13.1±8.1</td>
<td>6.8±3.9</td>
<td>12.8±13.2</td>
<td>4.3±8.4</td>
</tr>
<tr>
<td>winequality-white-3 vs 7</td>
<td>41.5±9.5</td>
<td>37.7±19.2</td>
<td>36.2±15.0</td>
<td>32.7±16.5</td>
</tr>
<tr>
<td>abalone-17 vs 7-8-9-10</td>
<td>28.7±7.9</td>
<td>21.4±7.5</td>
<td>22.2±10.2</td>
<td>23.8±7.6</td>
</tr>
<tr>
<td>kr-vs-k-three vs eleven</td>
<td>99.8±0.6</td>
<td>96.0±5.1</td>
<td>96.8±2.4</td>
<td>96.7±2.8</td>
</tr>
<tr>
<td>yeast5</td>
<td>67.2±8.2</td>
<td>62.8±16.8</td>
<td>67.6±4.6</td>
<td>62.6±13.4</td>
</tr>
<tr>
<td>winequality-white-9 vs 4</td>
<td>41.7±35.4</td>
<td>30.3±34.6</td>
<td>22.2±35.1</td>
<td>5.6±15.7</td>
</tr>
<tr>
<td>yeast-1-2-8-9 vs 7</td>
<td>29.9±12.1</td>
<td>22.2±13.6</td>
<td>25.4±14.8</td>
<td>21.2±16.7</td>
</tr>
<tr>
<td>poker-9 vs 7</td>
<td>35.1±17.1</td>
<td>25.4±18.7</td>
<td>24.1±23.0</td>
<td>15.4±20.2</td>
</tr>
<tr>
<td>car-vgood</td>
<td>99.9±0.2</td>
<td>97.3±5.0</td>
<td>96.4±4.2</td>
<td>83.2±31.7</td>
</tr>
<tr>
<td>glass-0-1-6 vs 5</td>
<td>71.2±28.9</td>
<td>65.7±32.4</td>
<td>56.3±34.4</td>
<td>36.7±35.5</td>
</tr>
<tr>
<td>zoo-3</td>
<td>35.3±29.9</td>
<td>29.4±21.4</td>
<td>32.2±30.0</td>
<td>20.4±29.2</td>
</tr>
<tr>
<td>abalone9-18</td>
<td>40.1±7.4</td>
<td>30.4±9.9</td>
<td>37.9±6.4</td>
<td>30.2±11.4</td>
</tr>
<tr>
<td>glass4</td>
<td>54.4±16.4</td>
<td>51.2±22.2</td>
<td>46.9±24.8</td>
<td>54.0±16.1</td>
</tr>
<tr>
<td>ecoli-0-1-4-6 vs 5</td>
<td>69.9±16.0</td>
<td>74.6±18.4</td>
<td>68.9±11.1</td>
<td>69.2±11.8</td>
</tr>
<tr>
<td>vowel0</td>
<td>94.7±5.2</td>
<td>97.7±2.1</td>
<td>89.4±5.8</td>
<td>91.9±4.5</td>
</tr>
<tr>
<td>yeast-0-5-6-7-9 vs 4</td>
<td>46.8±4.4</td>
<td>55.3±12.7</td>
<td>40.3±10.8</td>
<td>52.2±12.3</td>
</tr>
<tr>
<td>ecoli-0-1 vs 2-3-5</td>
<td>76.5±11.1</td>
<td>67.7±11.6</td>
<td>65.9±12.9</td>
<td>57.0±8.4</td>
</tr>
<tr>
<td>yeast-0-3-5-9 vs 7-8</td>
<td>42.1±8.3</td>
<td>36.9±11.5</td>
<td>29.4±6.9</td>
<td>29.1±11.8</td>
</tr>
<tr>
<td>yeast-2 vs 4</td>
<td>82.7±7.4</td>
<td>80.7±7.4</td>
<td>75.2±6.5</td>
<td>71.0±9.6</td>
</tr>
</tbody>
</table>
Experiments

In summary, over 24 datasets,

NLB wins 20 times in terms of AP
NLB wins 19 times in terms of F-score
Experiments

<table>
<thead>
<tr>
<th>Model</th>
<th>Average #Splits</th>
<th>Average #Weak learners</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB</td>
<td>22.13 ± 7.92</td>
<td>67.25 ± 35.55</td>
</tr>
<tr>
<td>NLB</td>
<td>5.08 ± 3.83</td>
<td>35.42 ± 39.01</td>
</tr>
</tbody>
</table>
During the experiments, we stored the number of splits per weak learner and the number of weak learner in average in the final model.

NLB uses 4 times less split and twice less weak learners, in average.

<table>
<thead>
<tr>
<th>Model</th>
<th>Average #Splits</th>
<th>Average #Weak learners</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB</td>
<td>22.13 ± 7.92</td>
<td>67.25 ± 35.55</td>
</tr>
<tr>
<td>NLB</td>
<td>5.08 ± 3.83</td>
<td>35.42 ± 39.01</td>
</tr>
</tbody>
</table>
Conclusion

- We presented a new Non-Linear Boosting Algorithm where we combine non-linearly different boosted learner
Conclusion

• We presented a new Non-Linear Boosting Algorithm where we combine non-linearly different boosted learner

• We showed how it can be suited for class imbalance problems
Conclusion

• We presented a new Non-Linear Boosting Algorithm where we combine non-linearly different boosted learner

• We showed how it can be suited for class imbalance problems

Future work:
• Adapt to multi-class setting
Conclusion

• We presented a new Non-Linear Boosting Algorithm where we combine non-linearly different boosted learner

• We showed how it can be suited for class imbalance problems

Future work:
• Adapt to multi-class setting
• Further study on the overfitting scenario
Thank you for your attention
BACKUP SLIDES
Linear Boosting Drawbacks

Linearly combining the models learner isn’t always optimal

The previous problem can be solved in \( \approx 8 \) stumps with non-linear combinations.
Linear Boosting Drawbacks

Linearly combining the models learner isn’t always optimal

The previous problem can be solved in \( \approx 8 \) stumps with non-linear combinations.
Linear Boosting Drawbacks

Weak learners are good for generalisation. However, linearly combining them isn’t optimal:

Solution?

- Learn stronger weak learner? (counterintuitive, high risk of overfitting)
- Allow different combination of the weak learner
Non-Linear Gradient Boosting

As in linear gradient boosting we need to find a new $h_t$:

$$h_t = \text{argmin}_h \sum_{i=1}^{M} L \left( \sum_{r=1}^{R} \alpha^r L_r (F_{t-1} + h(x_i)), y_i \right)$$

Find $\alpha^r_t$ such that:

$$\alpha^r_t = \text{argmin}_\alpha \sum_{i=1}^{M} L \left( \sum_{r=1}^{R} \alpha^r L_r (F_{t-1} + \alpha h_t(x_i)), y_i \right)$$

and update $\alpha^r$:

$$\alpha^r = \text{argmin}_\alpha \sum_{i=1}^{M} L \left( \sum_{r=1}^{R} \alpha L_r (F_{t-1} + \alpha^r_t h_t(x_i)), y_i \right)$$
Notes

- Pros:
  - Only works on the weak learners output space thus it is no more likely to overfit than GB.
  - Faster convergence rate

- Cons:
  - Computation complexity: $O(MTR)$ (against $O(MT)$ for GB)
  - Inefficient with strong base learner
Linear Boosting Drawbacks

The previous problem can be solved in \( \approx 8 \) stumps with non-linear combinations

How can we learn non-linear combinations?
Linear Boosting Drawbacks

The previous problem can be solved faster with non-linear combinations

How can we learn non-linear combinations?