On the intrinsic robustness to noise of some leading classifiers and symmetric loss function

An empirical evaluation
Introduction - Motivations and Framework

- Why?

- In this study, what is done?

- In this study, what is not done?
Fraud detection

Frauds:
Non authorized behaviours, security breaches, ...

Currently:
Fraudsters detected by experts advice and business rules

Automatic detection through supervised learning is critical:
mistakes are very dangerous
penalties inflicted to human actors: fairness, reputation of the group

Objective: Detecting fraudulent behaviour with confidence
Nature of Fraud data

Binary data:
Two classes only, positive and negative (PN)

Unbalanced data:
One class is outnumbered by the other

Data subject to corruption:
Observed examples may be mislabeled
What is done:

Study of the robustness of algorithm to noisy labels

- Tutorial value, **open source**
- Additional results to existing studies
- **Large** range of parameters, datasets, algorithms
- Exploration of **Symmetrical Loss**

- **Only** Noise Completely At Random
- **No** pre-filtering/cleansing
I - Noisy Data

II - Design of the Benchmark

III - Results
I - Noisy Data

II - Design of the Benchmark

III - Results
I - 2. What is noise?

**Binary classification:**
prediction function from observed individuals:

\[ \mathcal{X} \in \mathbb{R}^n, \mathcal{Y} \in \{+1, -1\} \quad f : \mathcal{X} \rightarrow \mathcal{Y} \]

**Noise:**
Anything that *obscures* the relationship between the features and class of an individual.
Errors, corruptions due to various sources.

**Attribute noise vs. Label noise**

Moderate impact on classification performances  
Important impact on classification performances

**Lack of information about the noising procedure, random process:**

\[ x, \hat{y} \in (\mathcal{X}, \mathcal{Y}), P(\hat{y} = +1 | y = -1) = \rho_{-1}, P(\hat{y} = -1 | y = +1) = \rho_{+1} \]
I - 3. Noise dependency
1 - 3. Noise dependency

[Diagram showing the effect of noise on a classifier.]
I - 3. Noise dependency

Virtual separator as a potential classifier.

- Positive example
- Negative example
- Noised example

Original dataset

- Noisy Completely At Random
  - Uniform noise
- Noisy At Random
  - More noise into negative class
- Noisy Not At Random
  - More noise near boundary
I - Noisy Data

II - Design of the Benchmark

III - Results
II - 1. Implementation

Python with jupyter notebook. Various machine learning libraries.

**Objectives** (maintainability, speed, quality):

- Store **precise** results
- Use **simple** data structures to store transformed datasets and results
- Benefit from computing power available: use **multiprocessing**
- **Keep track** of maximum informations
- Use **hard drive** rather than RAM
## II - 2. Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Columns</th>
<th>Numeric</th>
<th>Categorical</th>
<th>Avg modality</th>
<th>Rows</th>
<th>Cells</th>
<th>Minority %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>13</td>
<td>5</td>
<td>8</td>
<td>12,7</td>
<td>48842</td>
<td>634946</td>
<td>23,9</td>
</tr>
<tr>
<td>Bank</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>5,3</td>
<td>41188</td>
<td>823760</td>
<td>3,3</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td></td>
<td>699</td>
<td>6291</td>
<td>34,4</td>
</tr>
<tr>
<td>Eye State</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td></td>
<td>14980</td>
<td>209720</td>
<td>44,8</td>
</tr>
<tr>
<td>Mushroom</td>
<td>22</td>
<td>0</td>
<td>22</td>
<td>5,3</td>
<td>8416</td>
<td>185152</td>
<td>46,6</td>
</tr>
<tr>
<td>Phishing</td>
<td>68</td>
<td>68</td>
<td>0</td>
<td></td>
<td>11055</td>
<td>751740</td>
<td>44,3</td>
</tr>
<tr>
<td>Spambase</td>
<td>57</td>
<td>57</td>
<td>0</td>
<td></td>
<td>4601</td>
<td>262257</td>
<td>39,4</td>
</tr>
<tr>
<td>Trucks</td>
<td>170</td>
<td>170</td>
<td>0</td>
<td></td>
<td>76000</td>
<td>12920000</td>
<td>1,8</td>
</tr>
<tr>
<td>CM1</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td></td>
<td>498</td>
<td>10956</td>
<td>9,8</td>
</tr>
<tr>
<td>JM1</td>
<td>22</td>
<td>17</td>
<td>5</td>
<td>287,8</td>
<td>10885</td>
<td>239470</td>
<td>19,3</td>
</tr>
<tr>
<td>KC1</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td></td>
<td>2105</td>
<td>46310</td>
<td>15,4</td>
</tr>
<tr>
<td>KC2</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td></td>
<td>522</td>
<td>11484</td>
<td>20,5</td>
</tr>
<tr>
<td>KC3</td>
<td>40</td>
<td>40</td>
<td>0</td>
<td></td>
<td>194</td>
<td>7760</td>
<td>18,6</td>
</tr>
<tr>
<td>PC1</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td></td>
<td>1109</td>
<td>24398</td>
<td>6,9</td>
</tr>
</tbody>
</table>
II - 3. Recipe

D publicly available datasets (.csv, .arff, .data, …)

→ Preprocessing / Standardization (missing, column selection, Y binarization) : D x (X, Y) en .csv

→ stratified K fold repeated R times : D x K x R (Xtrain, Ytrain) (Xtest, Ytest)

→ Swap Ytrain labels with a ρ probability scaled to the balance : D x K x R x |ρ| (Xtrain, Ŷtrain)

→ Learning on each set created on A algorithms

→ Evaluation on (Xtrain, Ytrain) and (Xtest, Ytest) with metrics M

→ Résultats : D x K x R x |ρ| x A x M

14 * 10 * 5 * 5 * 12 = 42,000 learnt models
II - 4. Algorithms

Scikit-Learn :
- Linear SVC
- Logistic Regression
- Random Forest

Weka :
- Random Forest

Khiops :
- Base Khiops
- Khiops with Random Forest generated features

XGBoost :
- XGBoost with 4 distinct losses :
  hinge, squared error, unhinged, ramp
II - 5. Symmetric losses

Symmetric loss function if: \( l : \mathbb{R} \rightarrow \mathbb{R}, \ l(z) + l(-z) = K, \) \( K \) being constant

Asymmetric losses:

Symmetric losses:
II - 6. Metrics

Accuracy is a problem for imbalanced datasets

- **Balanced accuracy**
  Accuracy where each individual is weighted with respect to its class probability

- **Cohen’s Kappa**
  \[ p_0 \text{ observed agreement ratio} \]
  \[ p_e \text{ expected agreement if random decisions} \]
  \[
  Kappa = \frac{(p_0 - p_e)}{(1 - p_e)}
  \]

- **Area Under ROC Curve (AUC)**

I - Noisy Data

II - Design of the Benchmark

III - Results
III - 1. Overview

Results on Breastcancer averaged per metrics, algorithms and noise applied

General decrease of performances along with noise, collapse when noise > minority% : no information left
III - 2. Robustness

\[
\text{Retained Performance} = \frac{\text{Result}_\rho}{\text{Result}_{\rho=0}}
\]

Some algorithms are more stable, are they better?
III - 3. Robustness scenarios

Impact on performances along with noise addition on spambase

Impact on performances along with noise addition on adult
Conclusion
Conclusion

- Random Forest is the best option when low noise
- If noise is suspected to occur, Khiops looks a stable option
- SVM and LR despite being “simple” are quite trustworthy
- Symmetrical Loss look promising, not yet a worth option
Thank you

Questions?