Open Anti-Cheat System

An opensource anti-cheat system
for realtime online multiplayer games

By Stephen Larroque

Supervisor: Pierre-Henri Wuillemin

UPMC
SORBONNE UNIVERSITÉS

Outline

- Problematic
- Theoretical approach
- Implementation
- Experiments and Results
- Applications and conclusion

Problematic

of cheating in online games

Problematic

- Online games:
  - Are a big part of the videogame industry
  - Important source of revenues
- Major challenges:
  - Network management
  - Online cheating
- Problem: lots of cheat types and lots of cheat softwares for each type (new ones everyday)

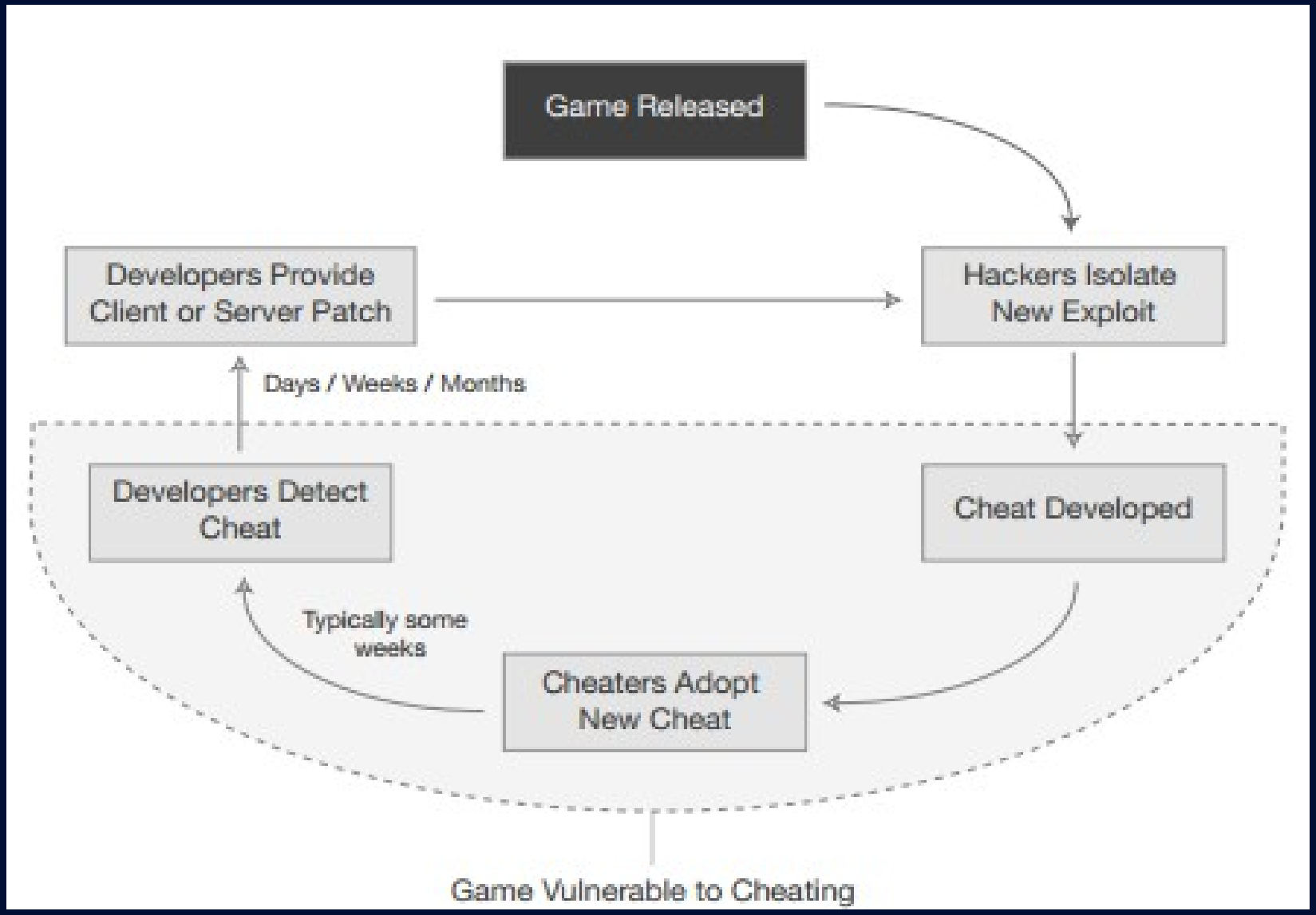
Taxonomy

- Using here the taxonomy of cheats by Yan & Choi (2002) extended by Tolbaru (2011):
  - 10:
    Cheating by modification of the game or data
  - 11:
    Cheating by exploiting bugs or design flaws
    (however there are other more efficient ways to fix this type of cheat by doing server-side processing and providing partial, bare minimum informations to clients)
  - Game-specific cheats

Current systems

- Client-side, implies:
  - Not robust (signatures or exploit patch)
  - Game and cheat specific solution
  - Security by obscurity
- Cheating costs, but anti-cheating too!
- Cheat-patch cycle nightmare
- Reactivity is very slow

Cheat-patch cycle



Cheaters vs developers

Cheaters	Developers
Wide communities of sharing	Isolated (patents, different systems)
Script-kiddies to professionals	Professionals only (requires high expertise, even for maintenance)
Act	React
Surprise advantage	Depend on cheaters
Very clever and skilled	Weakly formed and few practical solutions
Private hacks / commercial hacks (with continuous supports)	… (cannot do anything)

My ideal system?

- Server-side and preemptive (no need to get a hand on the cheat to detect it)
- Collaborative, open strategy (opensource)
- Taxonomy of features
- Generic for any game and algorithm
- Modular, interchangeables algorithms
- Must be reliable (none false positive, minimise false negative)

Theoretical approach

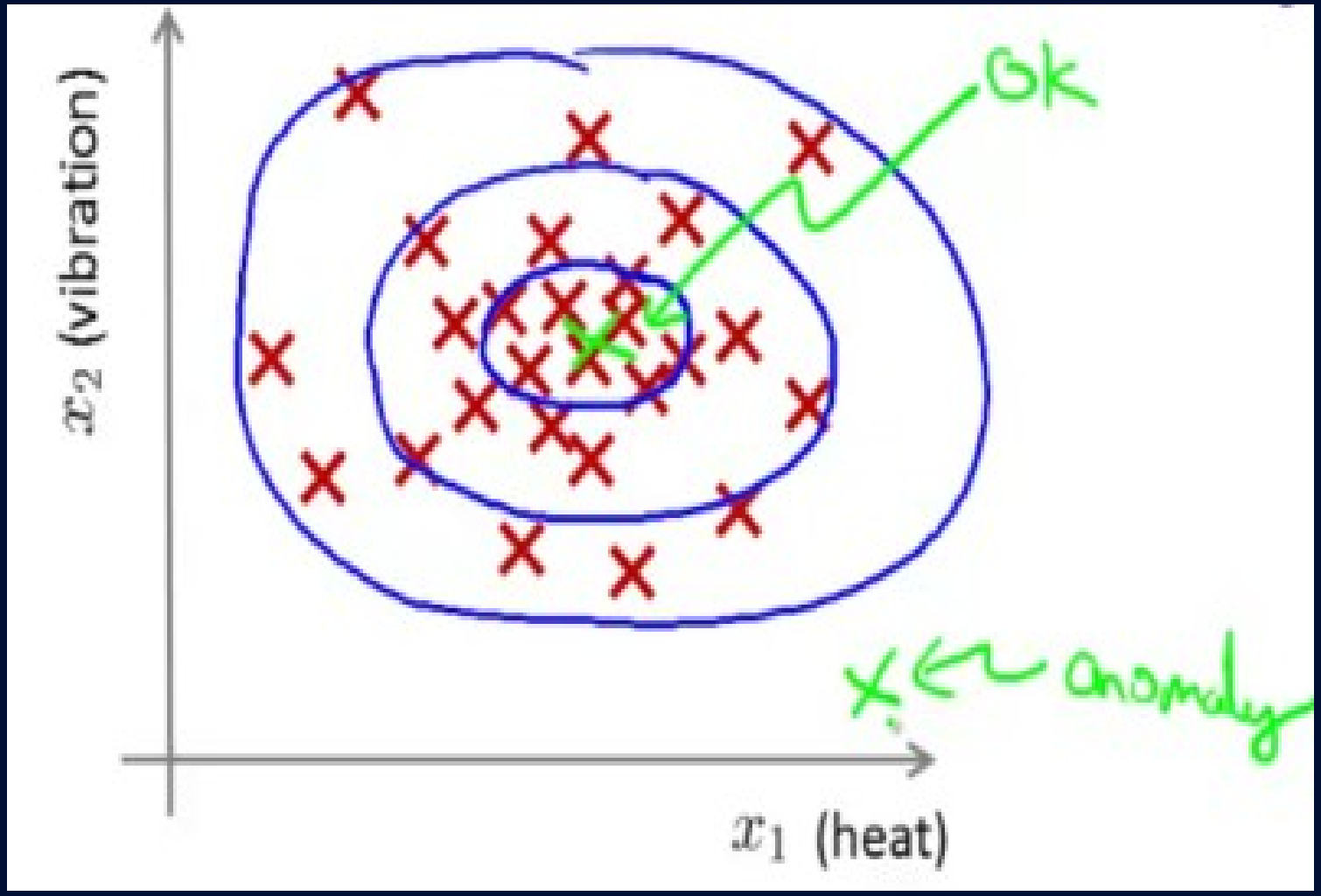
Behavioral analysis

- Lots of honest players
- Very few cheaters (relatively)
- Cheating behavior significantly different from honest
- Expert is able to discriminate
- A classifier can be the expert

Anomaly detection systems

- Family of semi-supervised classifiers
- Model over represented nominal behaviors (honest behaviors)
- Deviance = anomaly = potential cheat
- Semi-generative, semi-discriminative
- Similar to probabilistic classifiers, but only one unique class to learn

Anomaly detection systems

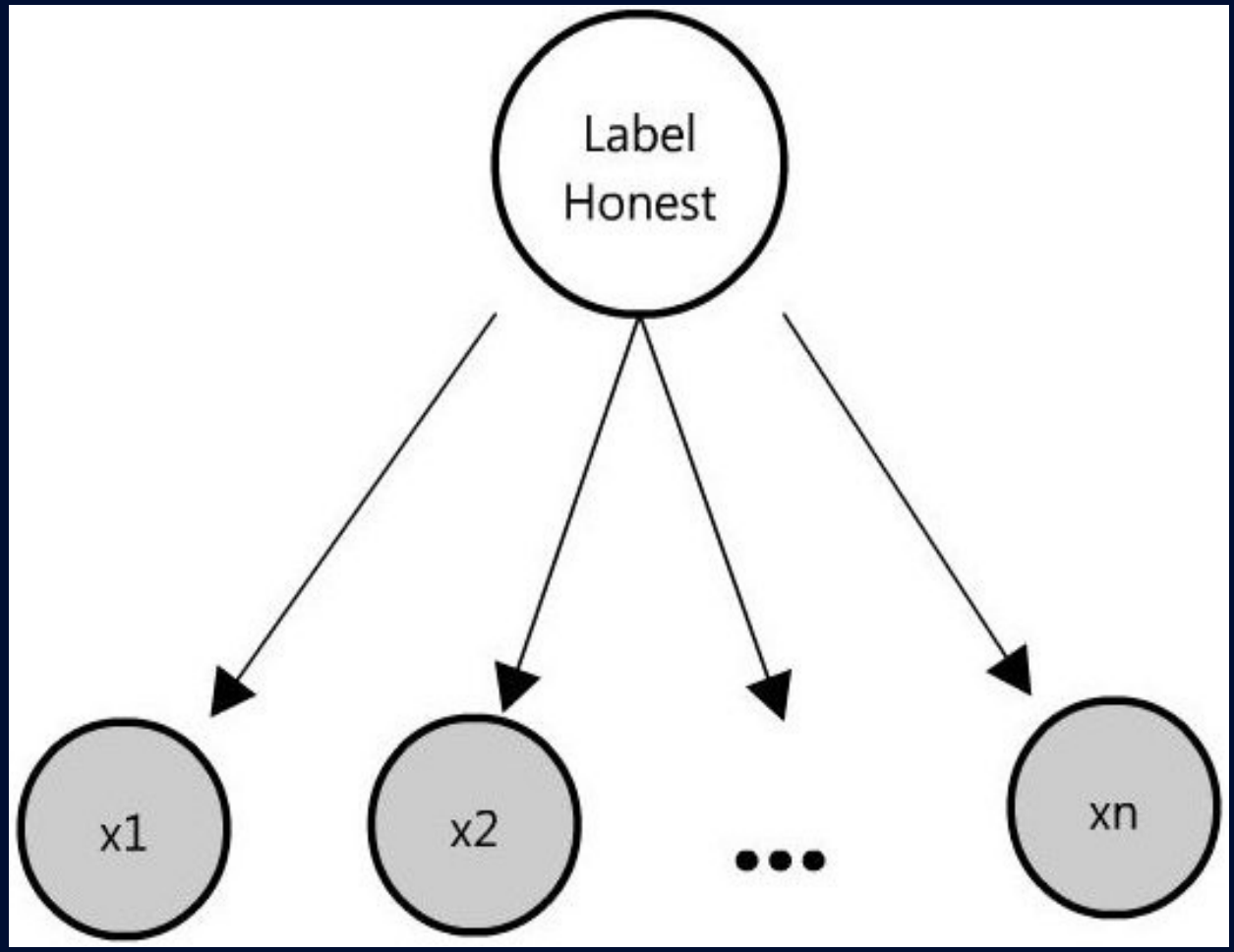


Anomaly detection systems

- Univariate Gaussian
- Multivariate Gaussian
- CAG
  Cluster-Augmented Gaussian

Univariate Gaussian

- I.I.D hypothesis (like Naive Bayes)



Univariate Gaussian - 2

- Learning: means and variances (vectors, for each feature)
- Detection: “likelihood” + Z threshold

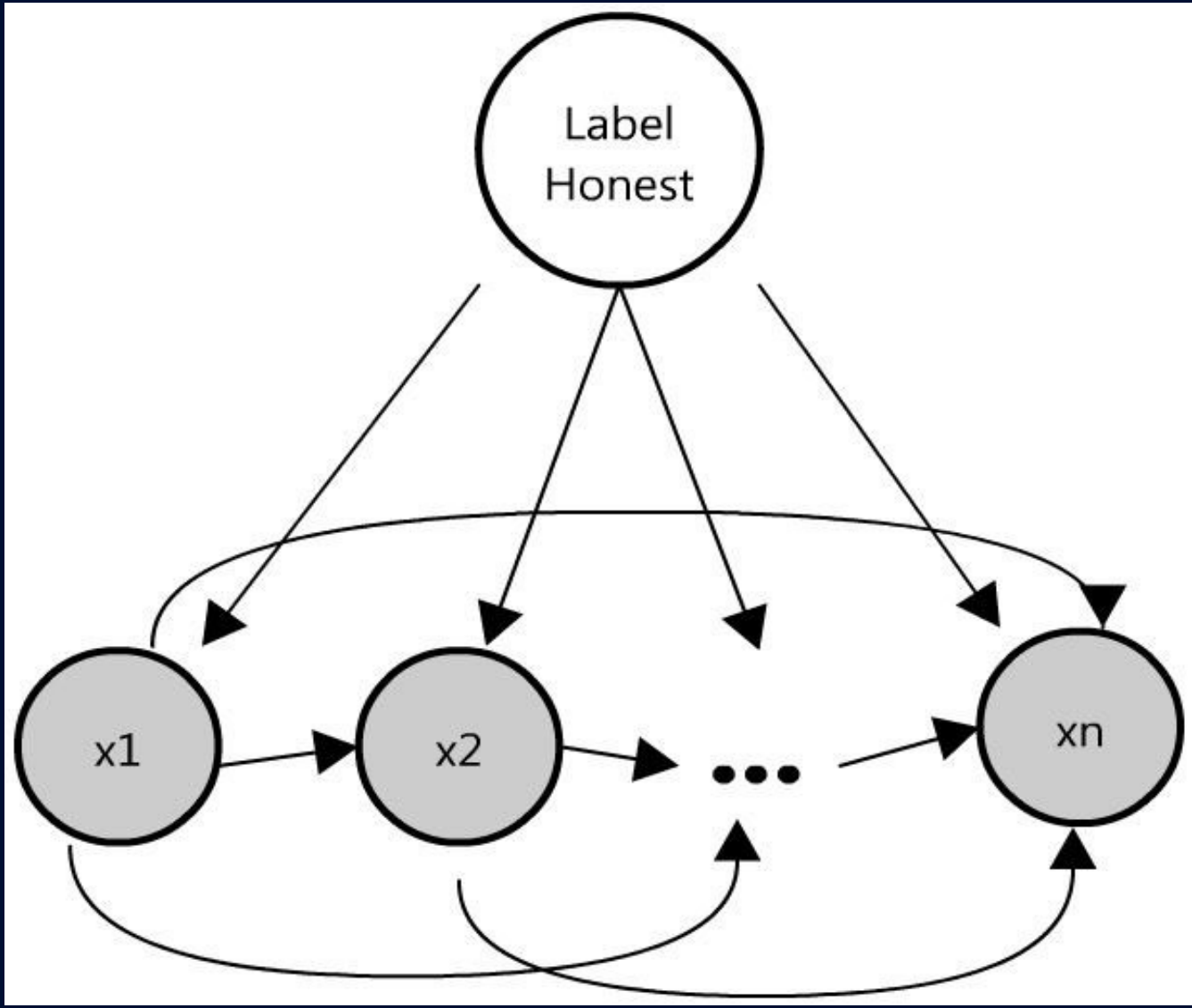
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

p(x) $\rightarrow$ 0 <=> cheater
p(x) < Z => potential cheater

Univariate Gaussian - 3

- Pros:
  - Very fast
  - Very simple to compute
- Cons:
  - Naive hypothesis
  - No correlation between features

Multivariate Gaussian



Multivariate Gaussian - 2

- Learning: means and covariance matrix

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

- Detection: “likelihood” + Z threshold

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

p(x) $\rightarrow$ 0 <=> cheater

p(x) < Z => potential cheater

Multivariate Gaussian - 3

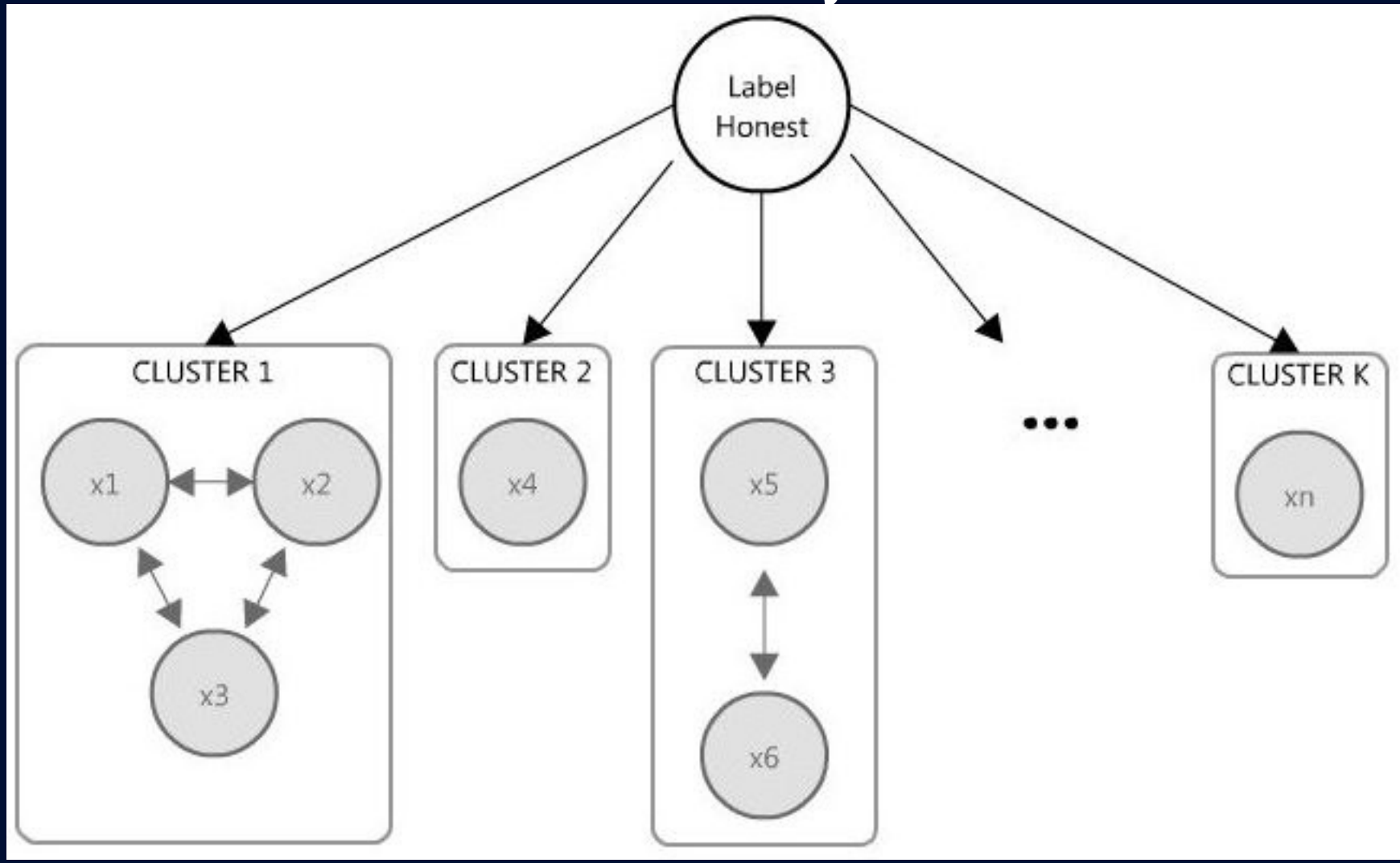
- Pros:
  - Capture all correlations
- Cons:
  - Complexity is quadratic in m (nb of features)

    Note: only at learning, at detection it's about constant time.

CAG – Our own algorithm

Cluster-Augmented Gaussian

- Features are clustered by max correlation



CAG - 2

- Learning:
  - Correlation by Mutual Information
  - Aggregating in clusters by “Kruskal-like” with fixed CMAX features per cluster
  - Mini covariance matrices for each cluster

CAG - 3

- Detection:
  - Multivariate gaussian “likelihood” using covar(k) (k=cluster index), if nb of features > 1
  - Univariate gaussian “likelihood” else.

CAG - 4

Exemple of clustering by CAG

Cluster 1
lastmouseeventtime
lastcommandtime
midair
movementdirection

Cluster 2
ducked
reactiontime
selfdamagecount
powerup_quad

Cluster 3
speedratio
cmdtime_reactiontime
weaponstate
angleinaframe

Cluster 4
scoreacc

CAG - 5

- Pros:
  - Capture most important correlations
  - May remove false correlations (noise)
  - Complexity is linear O(CMAX^2 * K) with K nb of clusters and for fixed CMAX
- Cons:
  - May “break” a few important correlations during clustering (because CMAX is reached for the current cluster)

Implementation

of our anti-cheat system

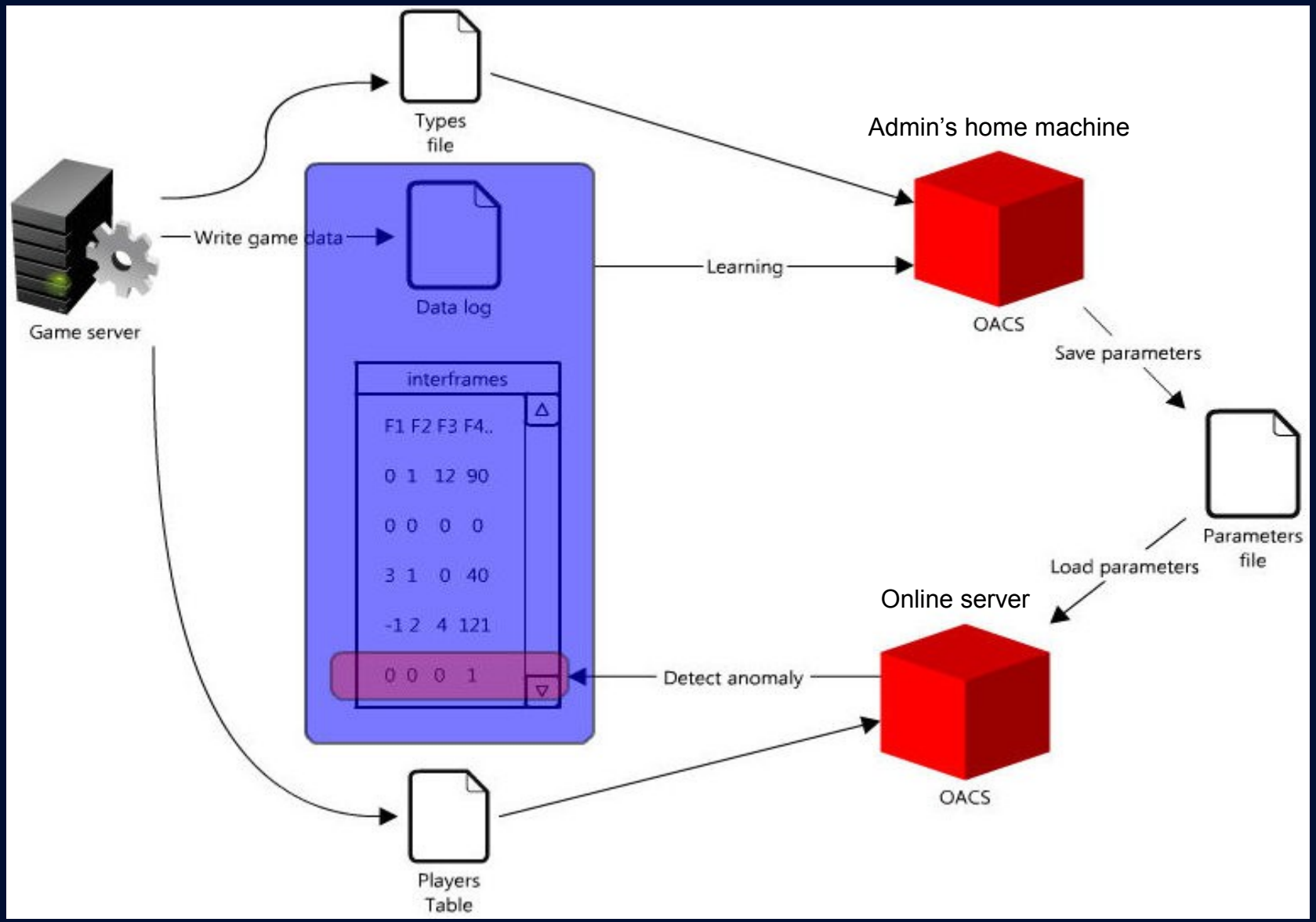
OACS

- Framework for anti-cheat development
- Modular (all is a module)
- Open strategy (opensource code, data and parameters are shareable)
- Robust : server-side, “un-hackable”
- Generic (any game, any algorithm)
- Reusable (requires only an interface)
- Coded in Python, powered by Pandas

OACS – Global mechanism

- 2 phases/modes:
  - Learning: learn parameters from a data file
  - Detection: continuously watch new entries in data file and report anomalies. Use previously learned parameters.

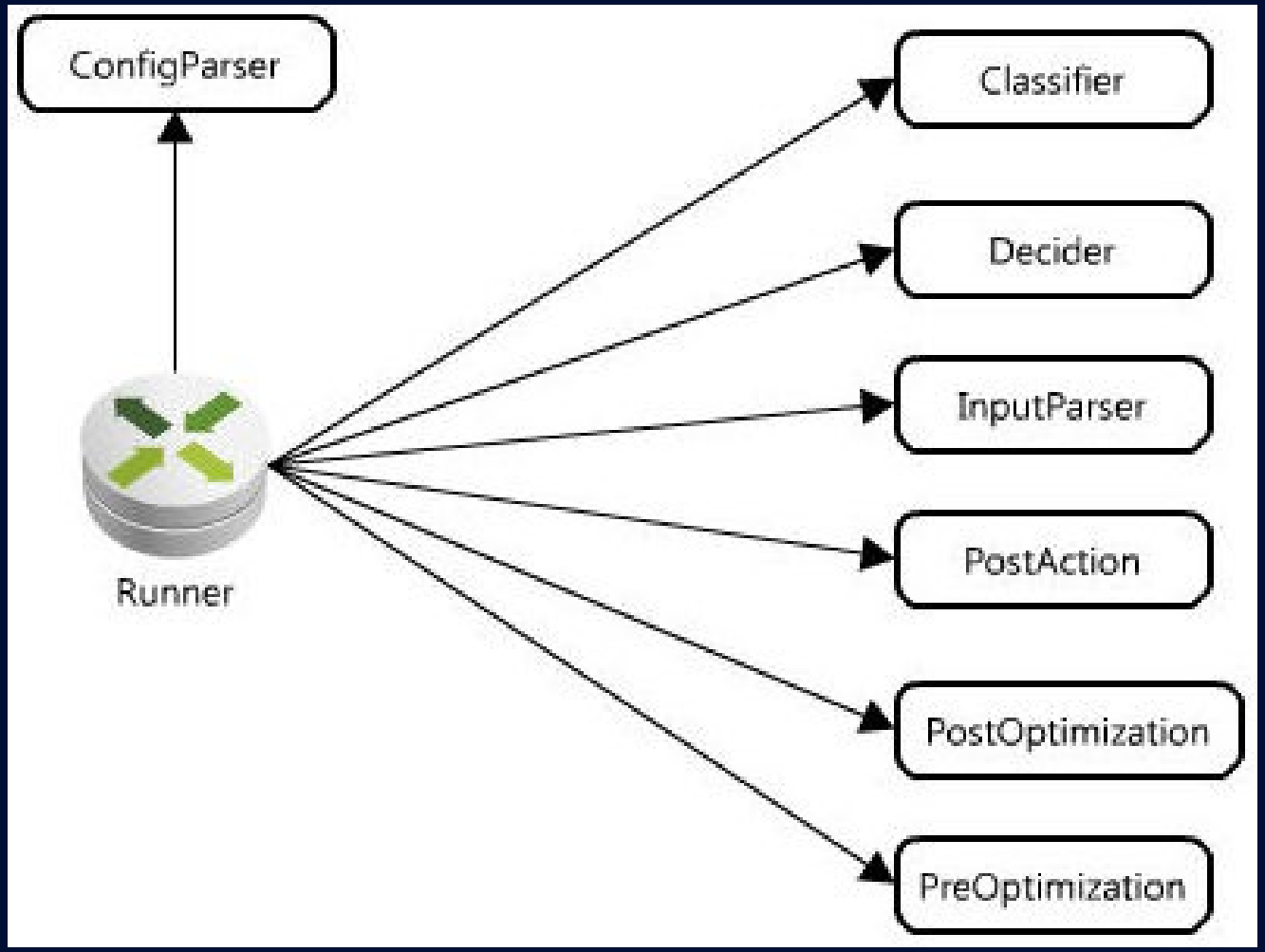
OACS – Global mechanism



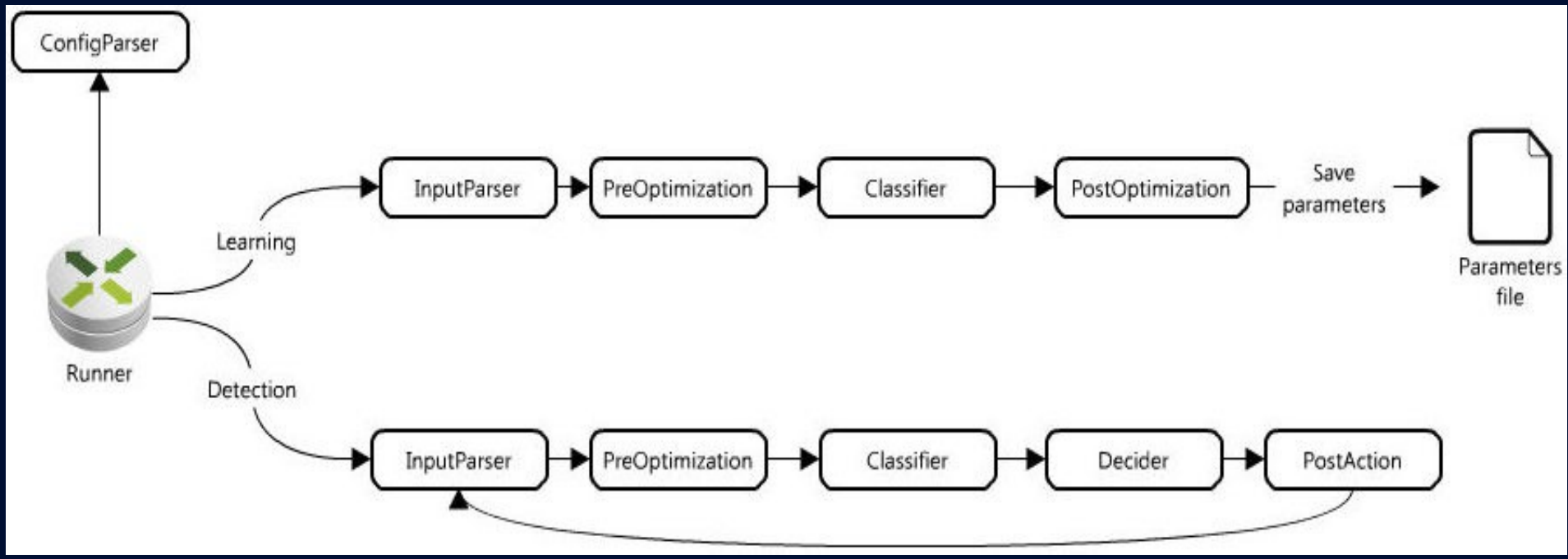
OACS – Internal mechanism

- Completely configurable (via config file and/or commandline arguments)
- All is module, classifiers too
- All modules are dynamically loaded
- Functions with generic and public interfaces (every modules can access any variable they need)

OACS – Internal mechanism

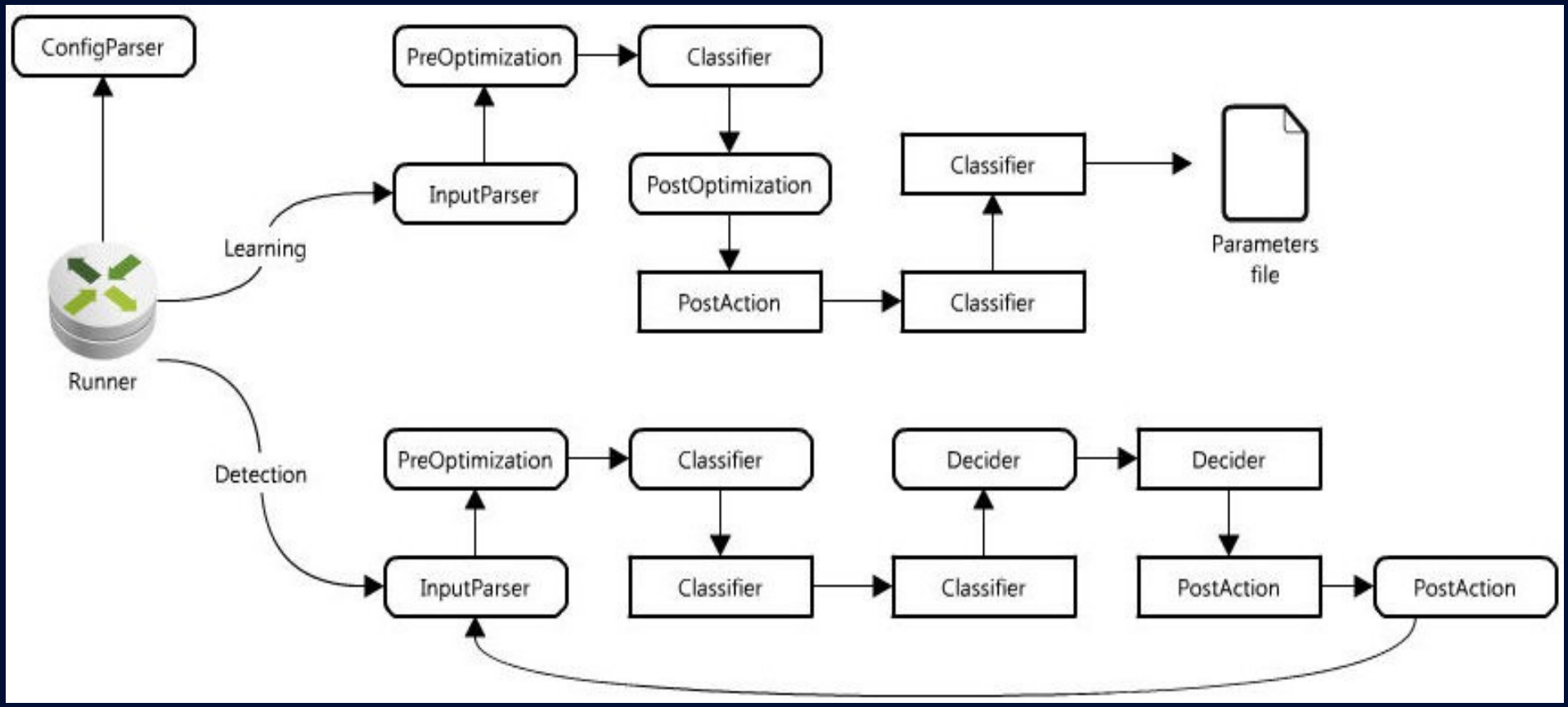


OACS – Workflow



Standard workflow

OACS – Workflow 2



Complicated custom workflow
with cascaded classifiers and deciders

OACS - Usage

- Learning mode:

```
python oacs.py --learn -c config.json --datafile
data.txt --typesfile types.txt -p parameters.txt
```

- Detection mode:

```
python oacs.py -c config.json --datafile data.txt
--typesfile types.txt -p parameters.txt -pt
playerstable.txt -dlog detectionlog.txt
```

- Possible to import and use OACS as a simple Python module!

OACS - GUI

- Interactive graphical interface with IPython Notebook

IP[y]: Notebook Dataset Analysis Last saved: May 09 4:09 PM

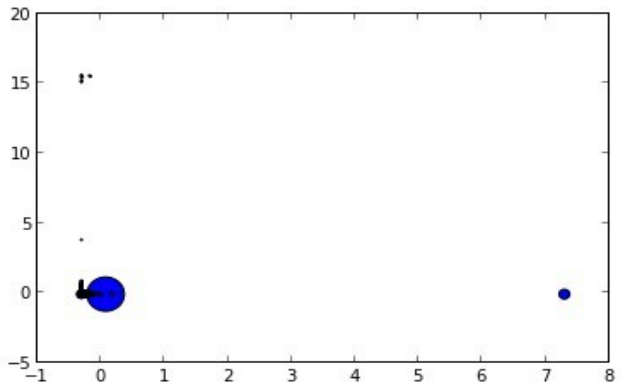
File Edit View Insert Cell Kernel Help

Markdown

1	0	16384	0	6	0.444	0	0
2	0	16384	0	6	0.000	0	0
3	0	16384	0	6	0.072	0	0
4	0	16384	0	6	0.403	0	0

In [31]:
```
# Printing angle in a frame in function of the reaction time and with the size
X = runner.vars['X']
plt.scatter(X.reactiontime, X.angleinaframe, s=X.framerepeat)
```

Out[31]: <matplotlib.collections.PathCollection at 0x8335da0>



Highly useful and advised to experiment and develop new modules!

Experiments and results

OpenArena

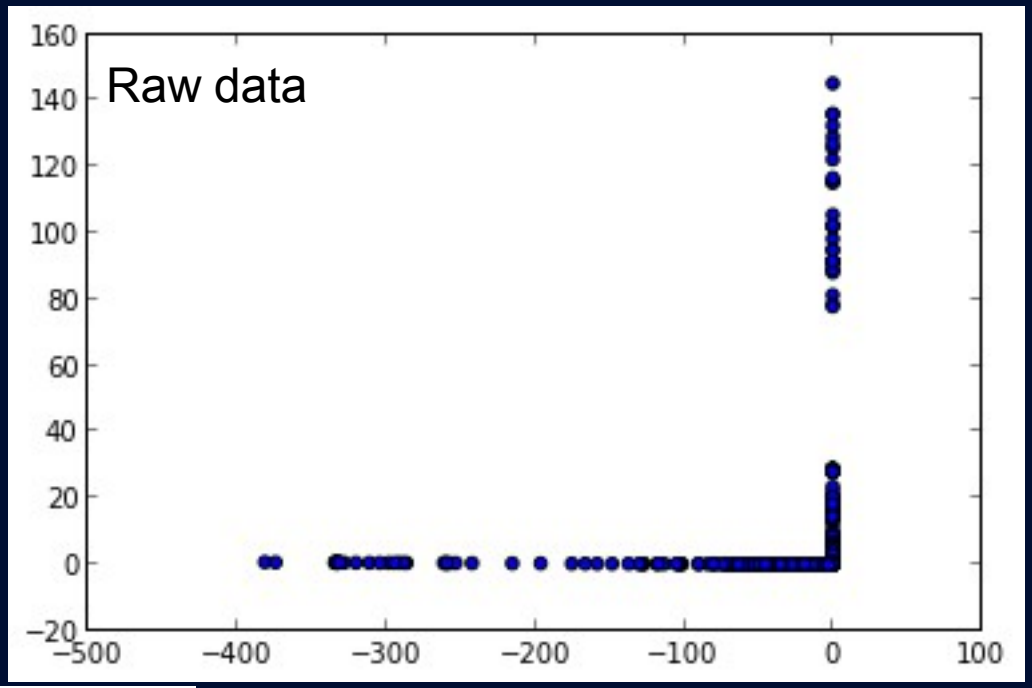
OpenArena - 2

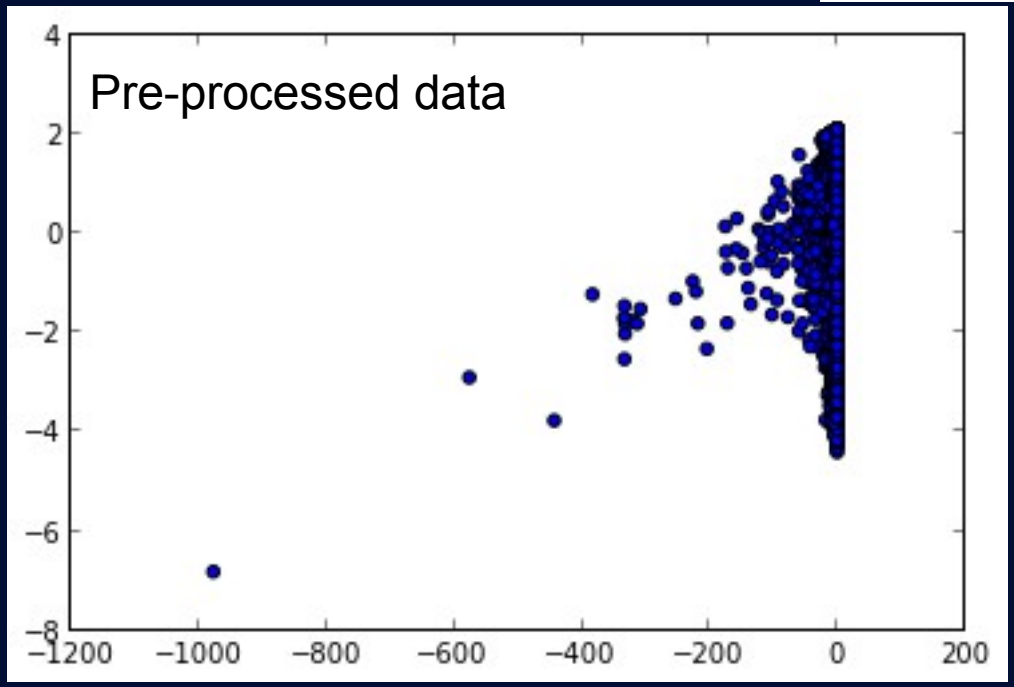
- Realtime 3D First-Person Shooter
- Based on ioquake3 (in C++)
- Code well documented
- Physics engine well known and analysed

Extended game server

- Interface between the game and OACS
- Record every player's actions
- In a **simple csv text file (this is the interface)**
- Action = Interframe = one line:
  difference between previous frame (world's state at instant t) and current frame
- Generic concept and adaptable to other games
- Easy to add other features

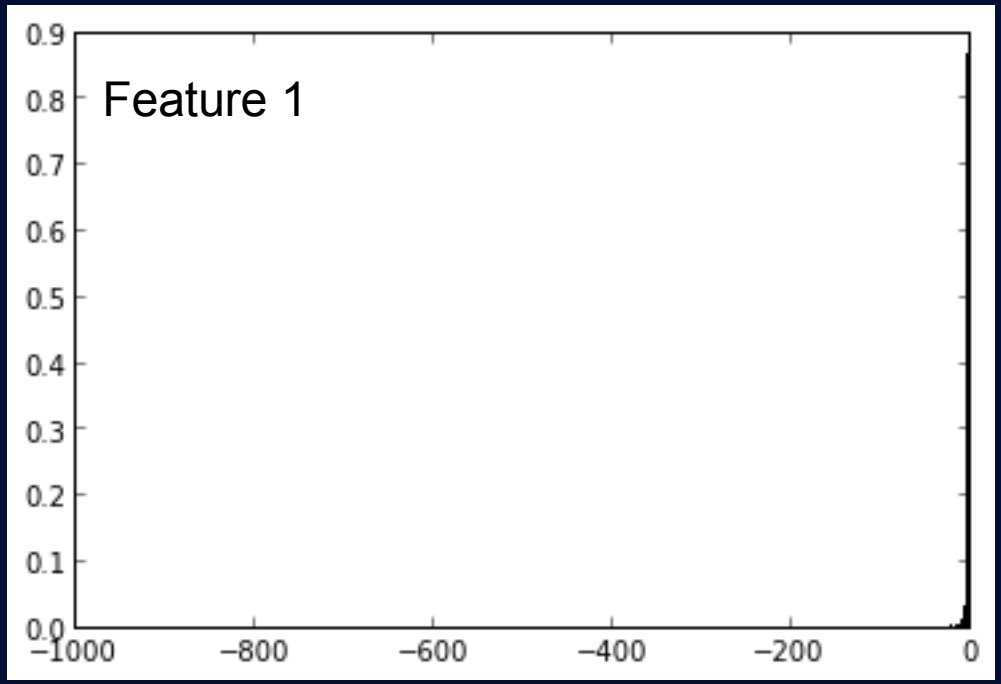
Results

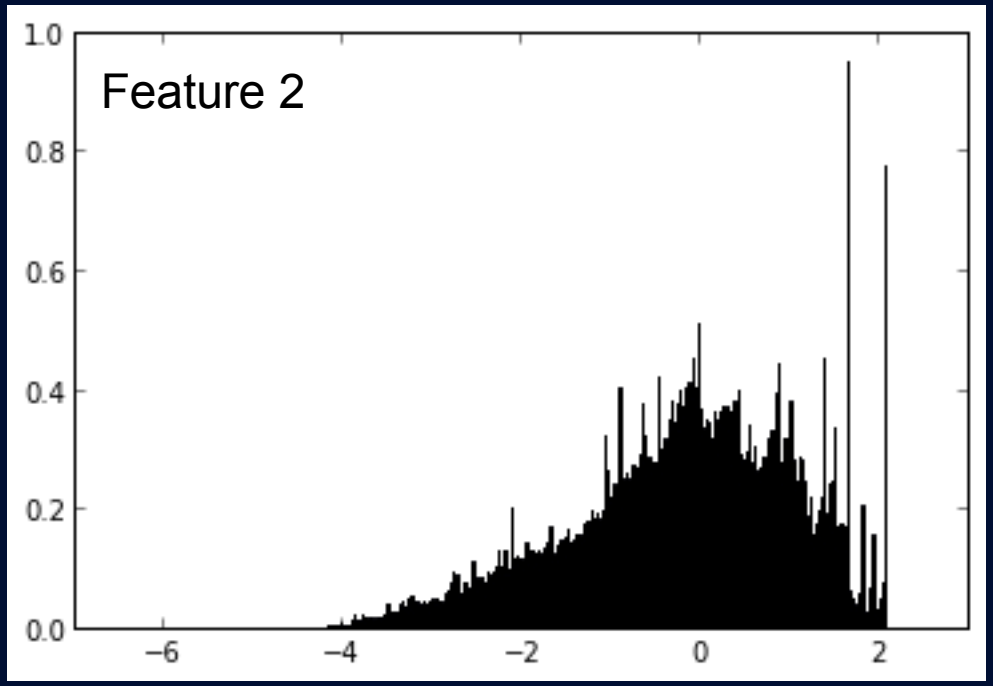

Raw data
160
140
120
100
80
60
40
20
0
−20
−500
−400
−300
−200
−100
0
100


Pre-processed data
4
2
0
−2
−4
−6
−8
−1200
−1000
−800
−600
−400
−200
0
200

Résultats 2

PCA dimensionality reduction in 2D

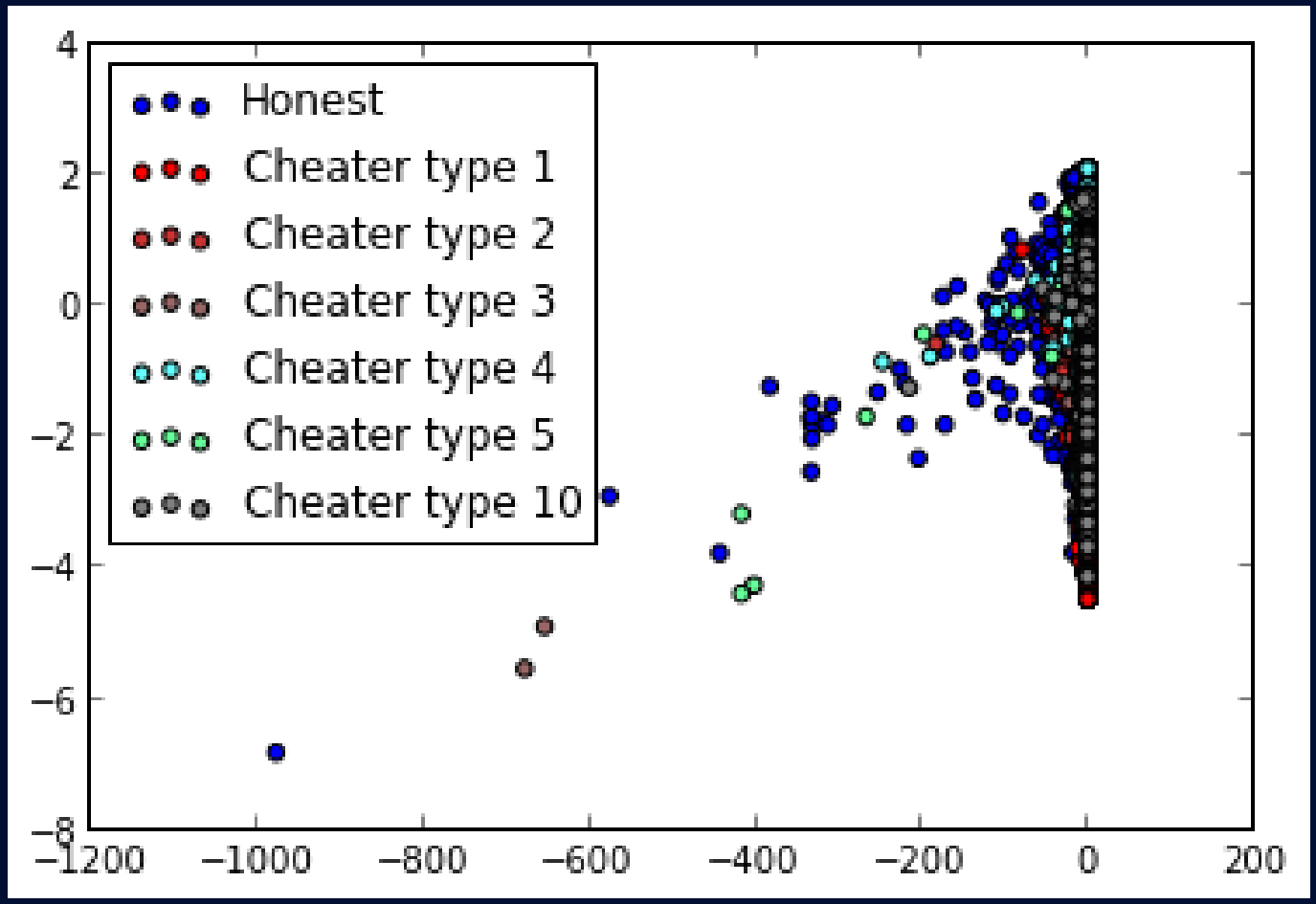




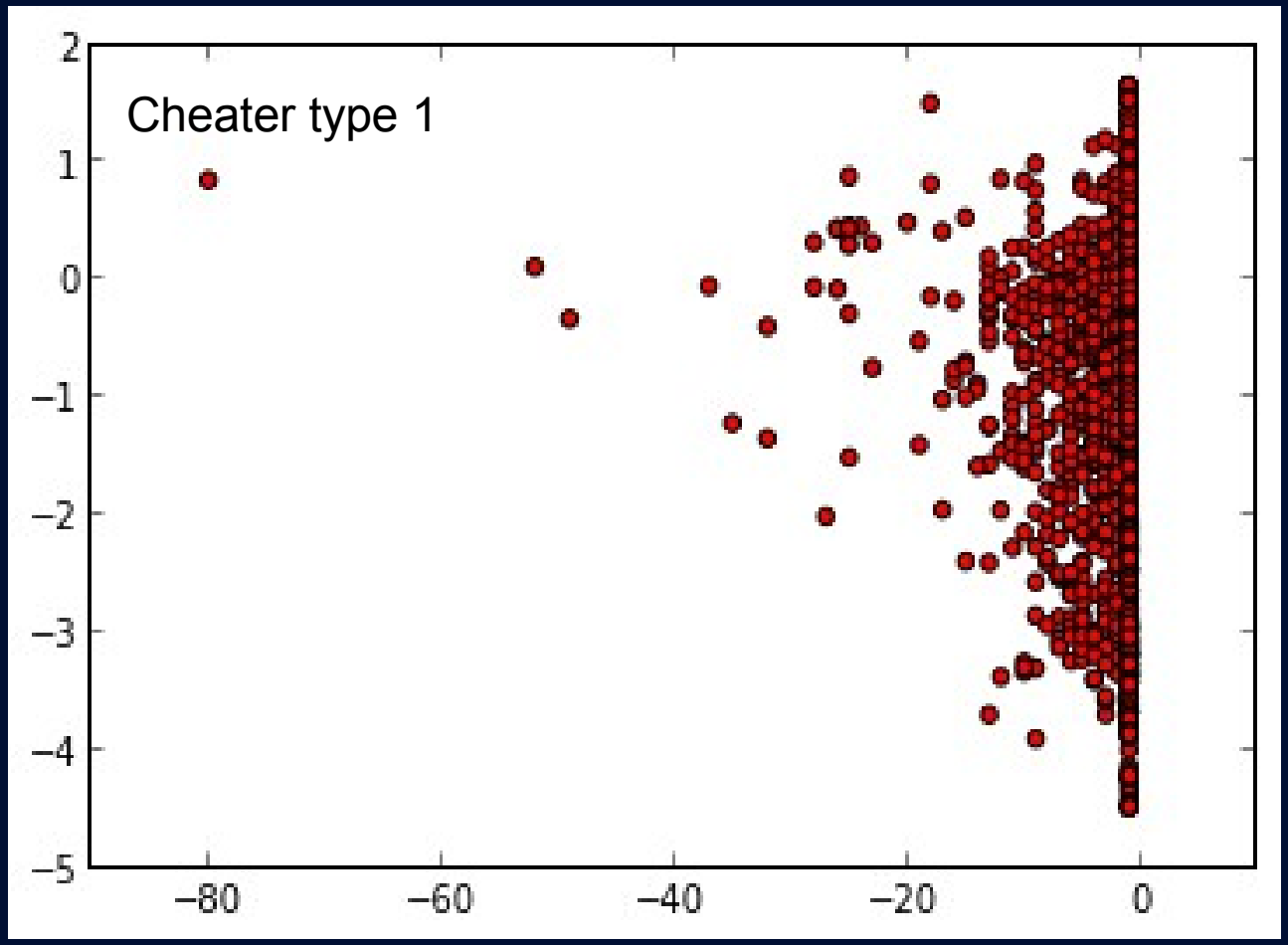
Feature 2 exhibits a gaussian curve, which is what we want!

PCA can be used to detect unusable features like in Feature 1.

Résultats 3

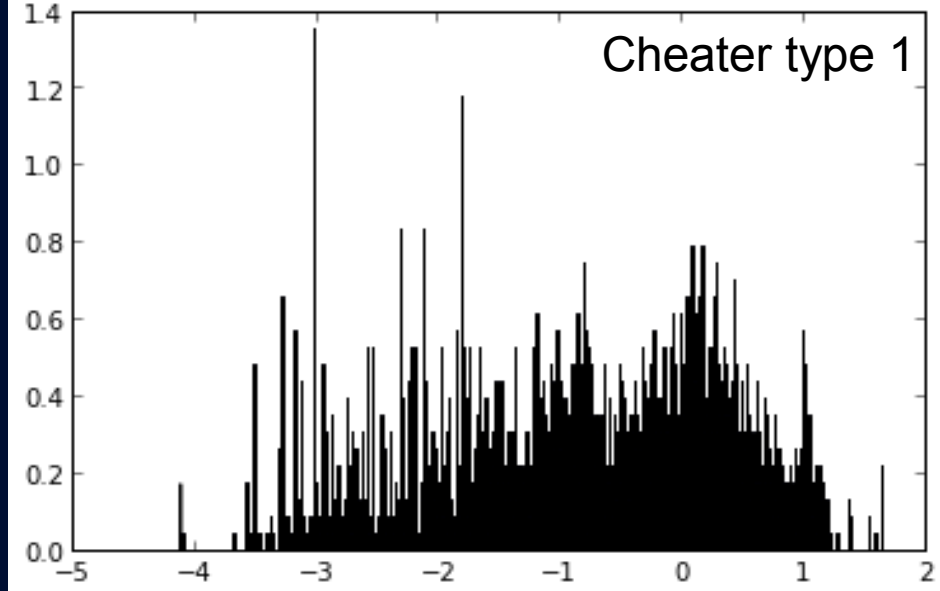

Honest
Cheater type 1
Cheater type 2
Cheater type 3
Cheater type 4
Cheater type 5
Cheater type 10
4
2
0
−2
−4
−6
−8
−1200
−1000
−800
−600
−400
−200
0
200

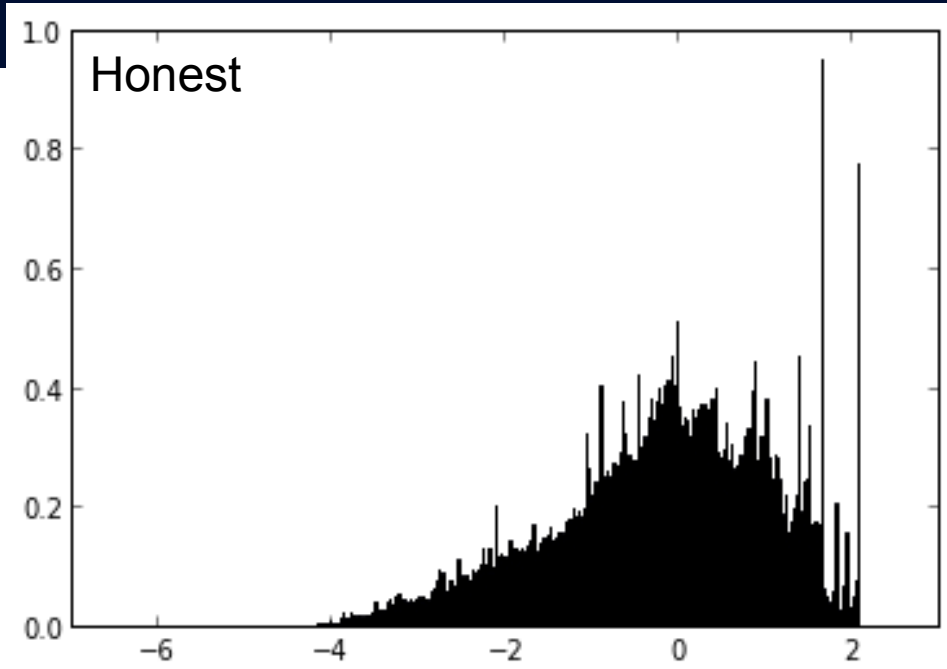
Résultats 4

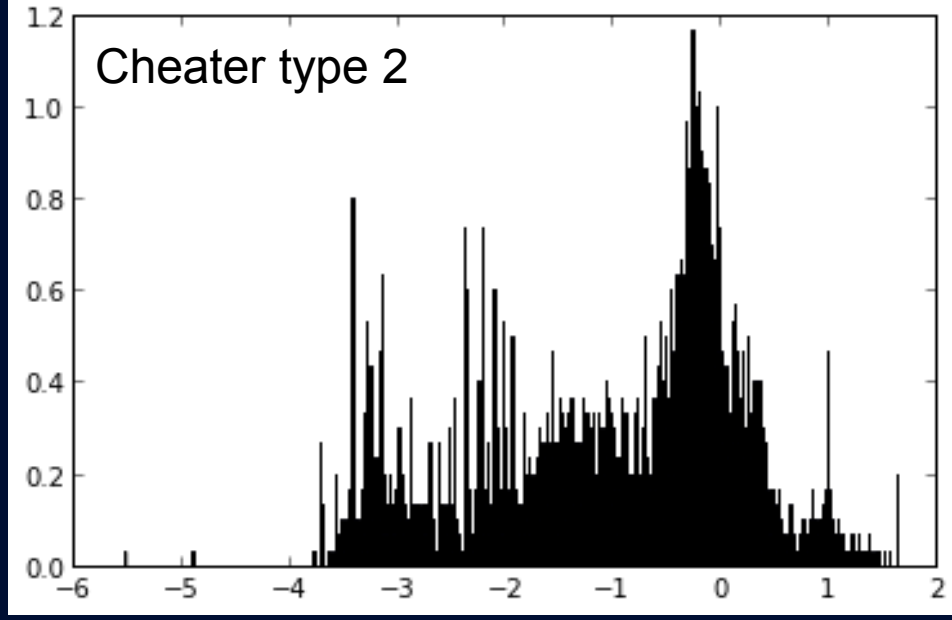


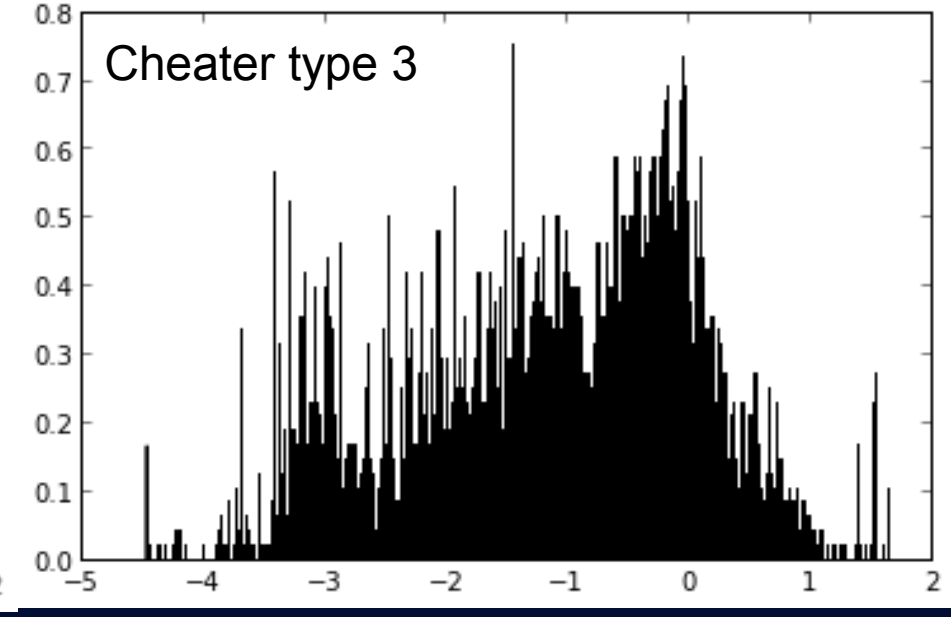
Résultats 5

Very different curves!

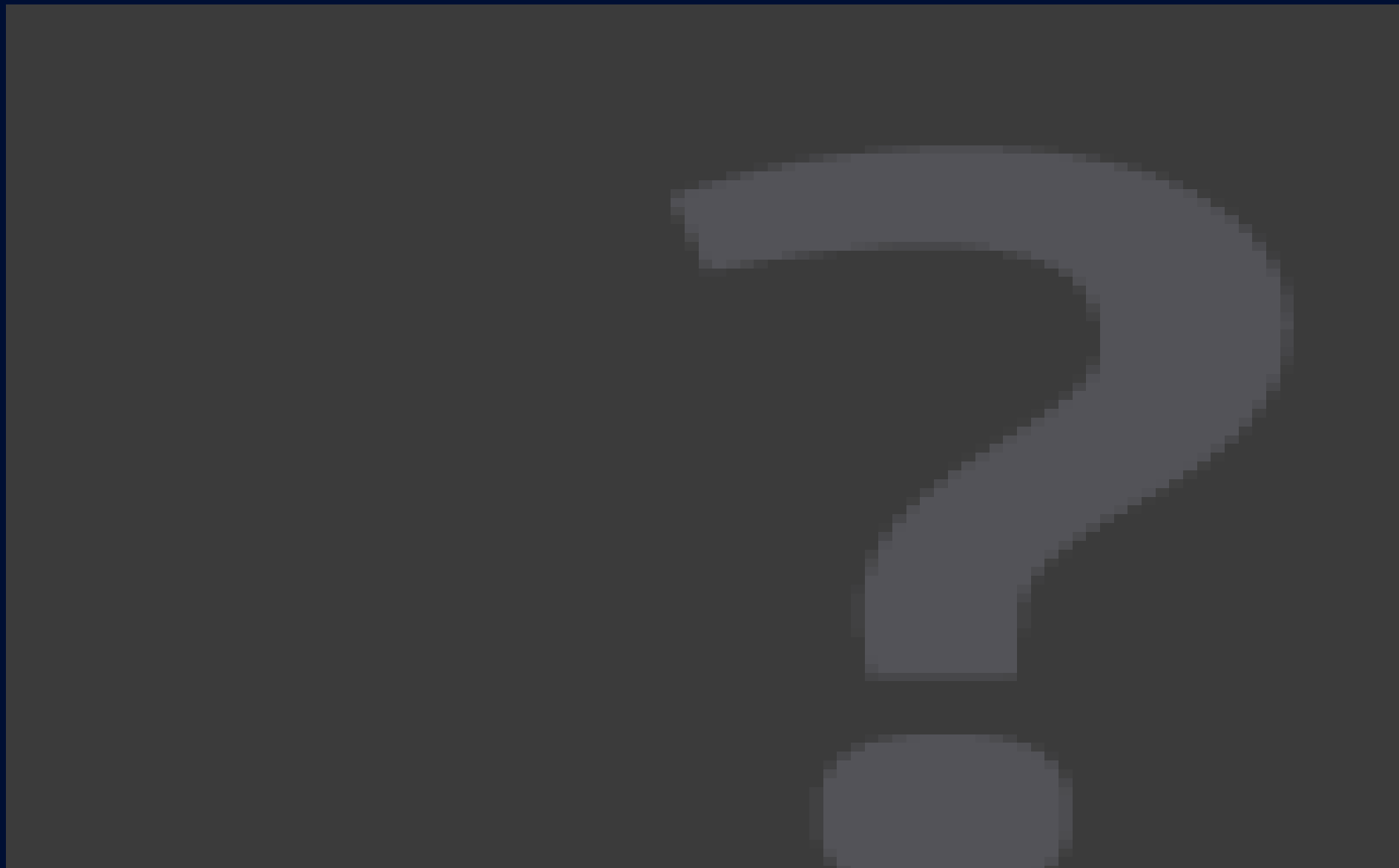








Résultats 6



Applications and conclusion

Applications and opening

- MMO* games
- Online casino games
- MOOC (online education classes)
- Alternative for CAPTCHAs
- Virtual economic systems and “gold duping” (BitCoins, WarCraft, SecondLife, etc..)
- Fraud detection and embezzlement (insurances, social funds, etc.)

Conclusion

- Fit to real data (honests >> cheaters)
- Robust to new cheat types and softwares
- Un-hackable (server-side)
- Generic, modular et reusable
- Reduce the “recoding” to only the data recording interface
- Model and data transformation to explore, but the framework already offers a good workspace

Thanking

- Andrew Ng
- Pierre-Henri Wuillemin
- Wes McKinney and Fernando Perez

References

- **Security Issues in Online Games**, J.J. Yan & H.J. Choi, 2002, The Electronic Library: international journal for the application of technology in information environments, Vol. 20 No.2
- **Cheating in Online Games - Threats and Solutions**, K. H. T. Morch, 2003, Norwegian Computing Center/Applied Research and Development, Publication No: DART/01/03.
- **Cheating in online video games**, Savu-Adrian Tolbaru, 2011, PhD Thesis
- **Self-Nonself Discrimination in a Computer**, S. Forrest and A. S. Perelson and L. Allen and R. Cherukuri,1994, in Proceedings of the 1992 IEEE Symposium on Security and Privacy.
- **Immunological Computation: Theory and Applications**, Dasgupta, Dipankar; Nino, Fernando (2008). CRC Press. p. 296. ISBN 978-1-4200-6545-9.
- **Fides: Remote Anomaly-Based Cheat Detection Using Client Emulation**, by Edward Kaiser, Wu-chang Feng, Travis Schluessler, November 2009

Bonus Slides

Features

- Features selection is crucial
- Some features are unusable and dangerous (high risk of false positive)
- How to choose the good ones?

Taxonomy of features

- 5 categories of features:
  - Identifiers (eg: playerid)
  - Human-specific (eg: reactiontime)
  - Game-specific (eg: score)
  - Physics-specific (eg: speed)
- Low level features (not aggregated, eg: no mean) and robust/invariant (human-specific)
- Accumulators (add a notion of temporality)

Interframes

Example of an interframe

playerid,timestamp,reactiontime,lastcmdtime,cmd_reactiontime,angleinaframe
385821367940981,1367940982,40,3620,38,3

- One line = One action
- Feature modifier = abstraction level + space disk reduction

Thank’s!