Carnegie Mellon University

amazon web services

蚂蚁金服 ANT FINANCIAL

Attributing Hacks

Yu-Xiang Wang

Joint work with

Ziqi Liu, Alex Smola, Kyle Soska, Qinghua Zheng




Amazon AI

Amazon AI

- Making machine learning and AI technologies accessible to all developers.

- We are hiring!
  - PhD Internship positions all year round.
  - Full-time positions also available.
  - Contact me, Anima, Alex or any other folks there.

Background

- There are 1,000,000,000 websites on the internet as of Sep 2014.
- About 1% of them are currently hacked or infected (source: securi.net)
- That's about 10 million malicious websites!

What can we do about it?

- Typically focus on detection and remediation.
  - Using small iFrames (Mavrommatis & Monrose, 08)
  - Norton Safe Web, McAfee Site Advisor.

- **Forensics / Attribution** of hacks
  - much harder problems
  - What? How? When?
  - This paper: use statistics, ML tools!

Outline

1. Challenges
2. Put ourselves in the hackers' shoes
3. Our solution: survival analysis + trend filtering
4. Results on real data

Challenge 1: hidden hacking procedure

Websites get hacked…

Whenever

- they are subject to a vulnerability (known to the attacker)
- they can be discovered efficiently
- the attacker has efficient tools

Knowledge

Opportunity

Tools

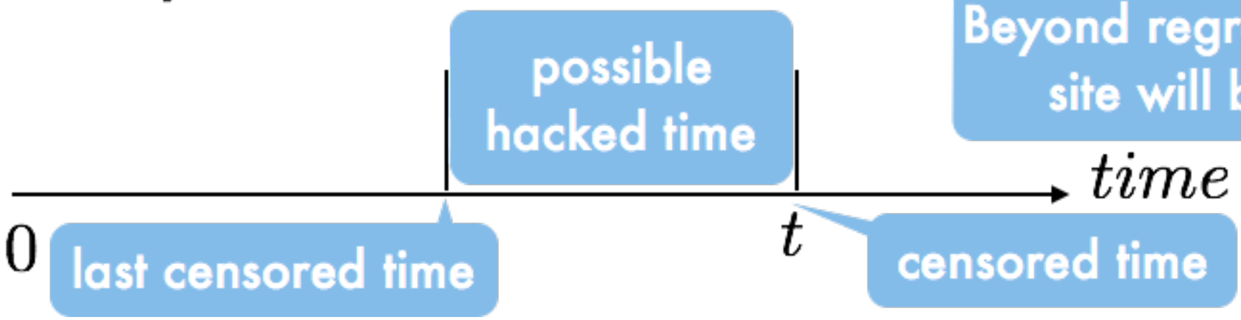
None of the three is known to us!

Challenge 2: Unknown hacking time

Uncertainty Do not know the exact time a site was hacked.

Observed The time ($t \in$ censored times) a compromised site will eventually be found and listed on the blacklists.

Beyond regression: when a site will be hacked?



No explicit labels for supervised learner.

Challenge 3: time varying risk

- Security risk is time sensitive.
  - Hackers keep discovering new exploits.
  - Websites keep patching bugs/vulnerability.
  - New versions of software are being installed.

Sharp changes triggered by events!

From a hacker's point of view

I found an exploit!
What to do?

Money?

Hack as many sites as possible
as quickly as possible

Fame?

Share the the exploit with peers.
Script kiddies will kick in.

CFEngine puppet labs PalletOps ANSIBLE CHEF SALTSTACK

What can we learn from this?

- Searchable string snippets are indicative features (Soska & Christin 2014)
  e.g., HTML tags <meta>WordPress 2.9.2</meta>
- Change points in hacking volume reveal hidden events/activities. (This paper!)

Outline

1. Challenges

2. Put ourselves in the hackers' shoes

3. **Our solution: survival analysis + trend filtering**

4. Results on real data

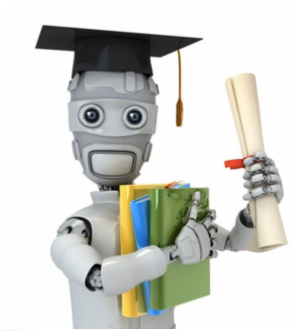
Recall the input and output

- Task: estimate the risk of getting hacked.

- Input:
  - Censored hack time.
  - features of websites.

- This is survival analysis!

Survival analysis



What the heck is that?

It's our bread and butter.

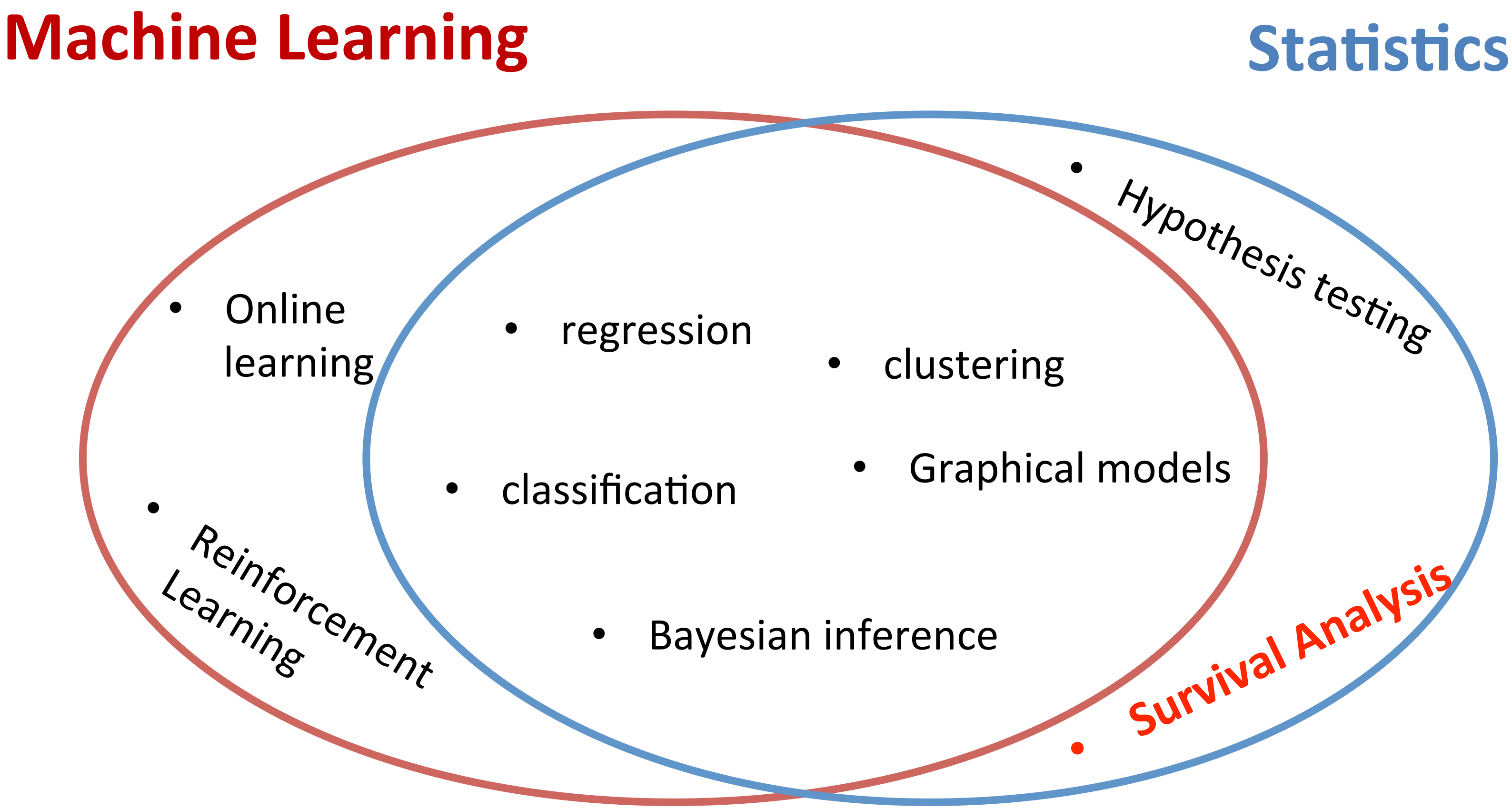
- Dates back to late 1600s, in studying smallpox and life expectancy.
- Still an active research area today.

Halley

Bernoulli

Modern formulation: (Kaplan & Meier, 1958; Cox, 1972)

- A density estimation problem for r.v. T: time of death.


Machine Learning
Statistics
Online learning
Reinforcement Learning
regression
clustering
classification
Graphical models
Bayesian inference
Hypothesis testing
Survival Analysis

Hacking as a survival problem

- A website got hacked ⇔ A patient had a heart attack.
- Vulnerable features ⇔ Genes assc. with heart disease
- Relay checkpoint ⇔ A regular physical checkup.
- Blacklisted ⇔ Diagnosed with heart failure

- Inferential tasks of interest:
  - Prob(Heart attack before age 40| DNA sequence x, healthy until 30)
  - Prob(hacked before May 1 | feature vector x, not hacked yet today)

Survival probability given x

$$F(T|x) = \exp\left(- \int_0^T \lambda(x,t)\, dt \right)$$

hazard rate: governs the probability of dying at time t if survive until t

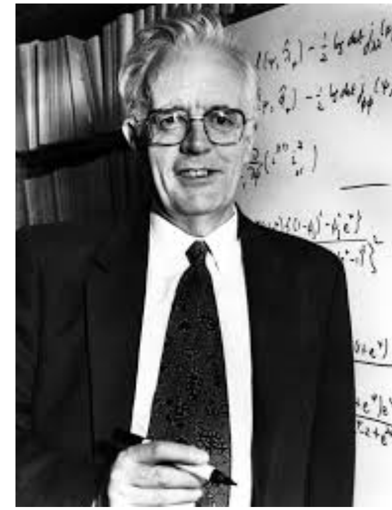
The Cox model

Cox (1972). "Regression models and Life-tables". Journal of the Royal Statistics Society.

$$\lambda(x, t; w) = \lambda_0(t) \exp \langle w, x \rangle$$

Parametric

Nonparametric, need to specify a parametric model



Sir David Cox

- A semi-parametric model.
- The "default" survival analysis model…
- Cited 44903 times (Google Scholar)!

From Cox model to our model

- Cox model: $\lambda(x, t; w) = \lambda_0(t) \exp \langle w, x \rangle$
  – Low dimensional generalized linear model

- Our model: $\lambda(x, t) = \langle x(t), w(t) \rangle$
  – Time varying, additive hazard function.
  – High dimensional. w is a vector of functions in t.
  – Fully nonparametric for each feature.

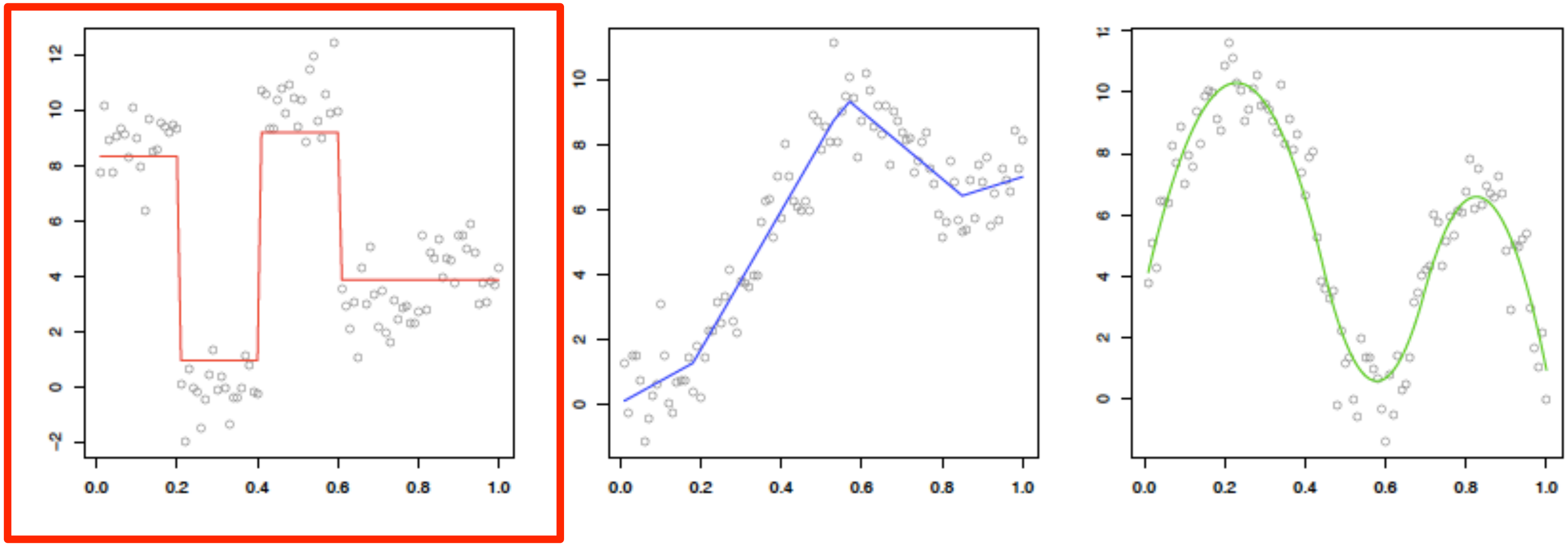
Comparing to existing time-varying survival models

- Kernel, smoothing splines (Kooperberg'94; Sauerbrei' 07)
  - Curse of dimensionality.
  - Require homogeneous smoothness.

- How we are doing differently?
  - Additive in each dimension.
  - Use trend filtering (Kim et. al.,, 2009; Tibshirani, 2013) to handle heterogeneous smoothness / sharp changes.

Locally adaptive nonparametric regression via trend filtering

Fused lasso



- For functions with bounded variation:
  - Trend Filtering: n^(-2/3) minimax rate
  - All linear smoothers: n^(-1/2) suboptimal rate

(Kim et. al. 2009, SIAM Review), (Tibshirani, AoS 2013), (W., Smola and Tibshirani, ICML'14)

Learning by regularized MLE

interval censoring

right censoring

$$\min_{(w_0,w_1,\ldots,w_p)\in\mathcal{F}^p} -\log\left[\prod_{i\in\mathcal{B}} p(t_i \le \tau_i < T_i) \prod_{i\notin\mathcal{B}} p(\tau_i > T)\right] + \gamma\sum_{j=0}^{p}\mathrm{TV}(w_j)$$

Nonnegative.
(Optionally) monotone.

allow for sharp changes

- Technical challenges:
  - This is optimizing over functions!
  - Interval censoring loss is non-convex
  - TV operator is non-smooth.

Our contributions

- Functions => Vectors in Euclidean space
  - The solution is parameterized by a small number of step-functions. (a cute re-parameterization and use of Mammen & Van De Geer, 1997)
- Handling non-smoothness via proximal SVRG.
  - Combine linear time proximal map using dynamic programming (Johnson, 2013) with results in (Yu, 2014)
  - Convergence rate despite non-convexity (Reddi et. al., 2016)
- Efficient implementation.
  - Represents only active sets.
  - Highly scalable, up millions of features and data points.

Key step of the prox-SVRG algorithm

- Doubly robust estimation
- Control variate.

(a) Pick a random minibatch $\mathcal{S} \subset [n]$:

$$w_j^{\mathbf{tmp}} = w_j^{(t)} - \eta \left(\sum_{i \in \mathcal{S}} \nabla g_i(w_j^{(t)}) - \boxed{\sum_{i \in \mathcal{S}} \nabla g_i(\tilde{w}_j) + \tilde{\mu}_j} \right).$$

(b) Solve the proximal map:

$$w_j^{(t+1)} = \begin{cases} \operatorname{argmin}_{w \in \mathbb{R}^{|\mathcal{T}|}} \frac{1}{2}\|w - w_j^{\mathbf{tmp}}\|^2 + \gamma\|Dw\|_1 + \delta(w \geq 0), & \text{for } \textbf{standard model}. \\ \operatorname{argmin}_{w \in \mathbb{R}^{|\mathcal{T}|}} \frac{1}{2}\|w - w_j^{\mathbf{tmp}}\|^2 + \gamma\|Dw\|_1 + \delta(w \geq 0) + \delta(Dw \geq 0), & \text{for } \textbf{monotone model}. \end{cases}$$

Stationarity convergence rate:

$O(n + n^{2/3}/\varepsilon)$ (Reddi et. al., 2016. Allen-Zhu, 2016.)

Proximal decomposition

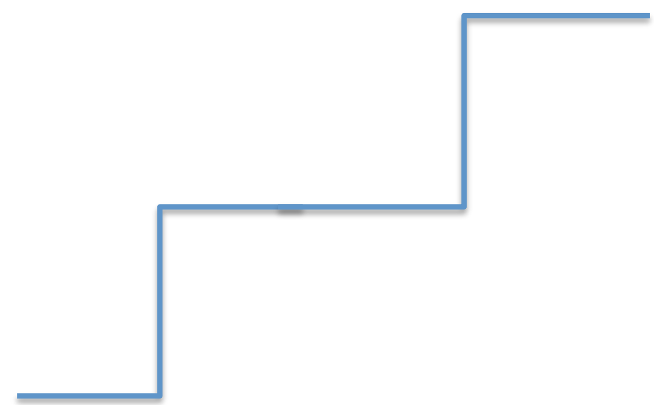
- Johnson (2013)'s DP algorithm solves:

$$w_j^{(t+1)} = \underset{w}{\operatorname{argmin}} \|w^{\boldsymbol{tmp2}} - w\|_2^2 + \gamma \|Dw\|_1$$

- But how to deal with the non-negativity?
  – Using Yaoliang Yu (2015)'s general characterization, we show that it decomposes!

TV penalty is not sensitive to sparsity.

- Do not distinguish between:



More sparsity (less bias) with TV-log

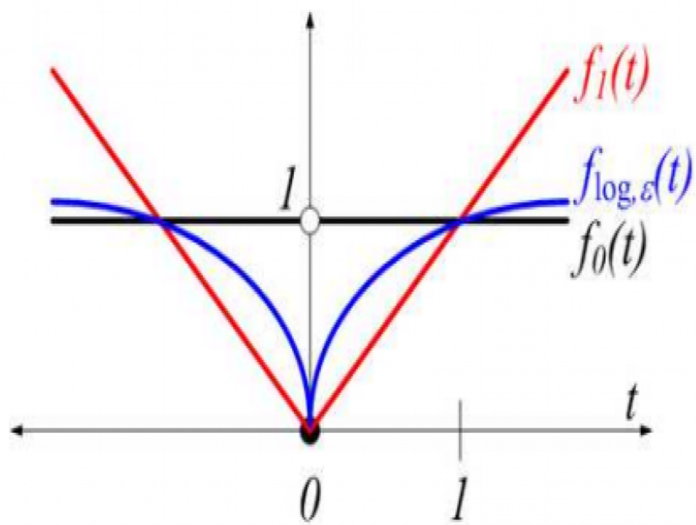


Figure 3: *At the origin, the canonical ℓ_0 sparsity count $f_0(t)$ is better approximated by the log-sum penalty function $f_{\log,\epsilon}(t)$ than by the traditional convex ℓ_1 relaxation $f_1(t)$.*

More sparsity (less bias) with TV-log

$$\mathrm{TV}(f) = \sup_{\mathcal{P}\in\left\{P=\{t_0,\ldots,t_{n_P}\}\,\middle|\,P \textit{ is a partition of } [a,b]\right\}} \sum_{i=1}^{n_P-1} |f(t_{i+1}) - f(t_i)|.$$

$$\mathrm{TV}_{\log}(f) := \sup_{\mathcal{P}\in\left\{P=\{t_0,\ldots,t_{n_P}\}\,\middle|\,P \text{ is a partition of } [a,b]\right\}} \sum_{i=1}^{n_P-1} \log(\epsilon + |f(t_{i+1}) - f(t_i)|).$$

Lemma 2. *For any function f we have that $\mathrm{TV}_{\log}(f) \leq \mathrm{TV}(f)$. Moreover, if f is Lipschitz continuous it follows that $\mathrm{TV}_{\log}(f) = \mathrm{TV}(f)$.*

For piecewise constant functions, TV_log is strictly smaller!

A novel variational definition.

How do we optimize it?

- Discrete TV_log = Discrete TV + Concave
- The concave part can be shown to be continuously differentiable.
- Combine the concave part with the loss functions. The same proximal SVRG!

Outline

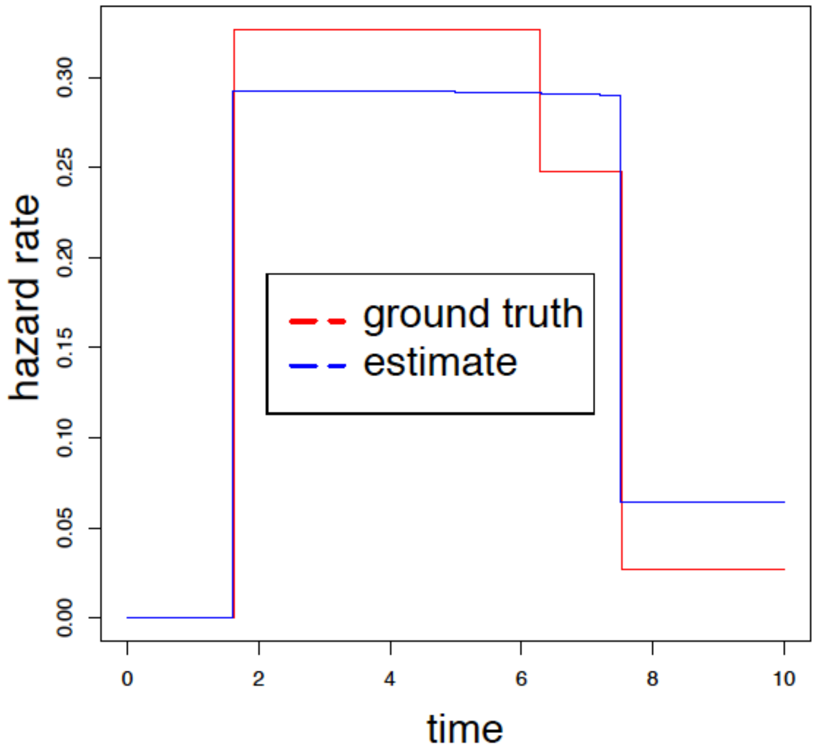
1. Challenges

2. Put ourselves in the hackers' shoes

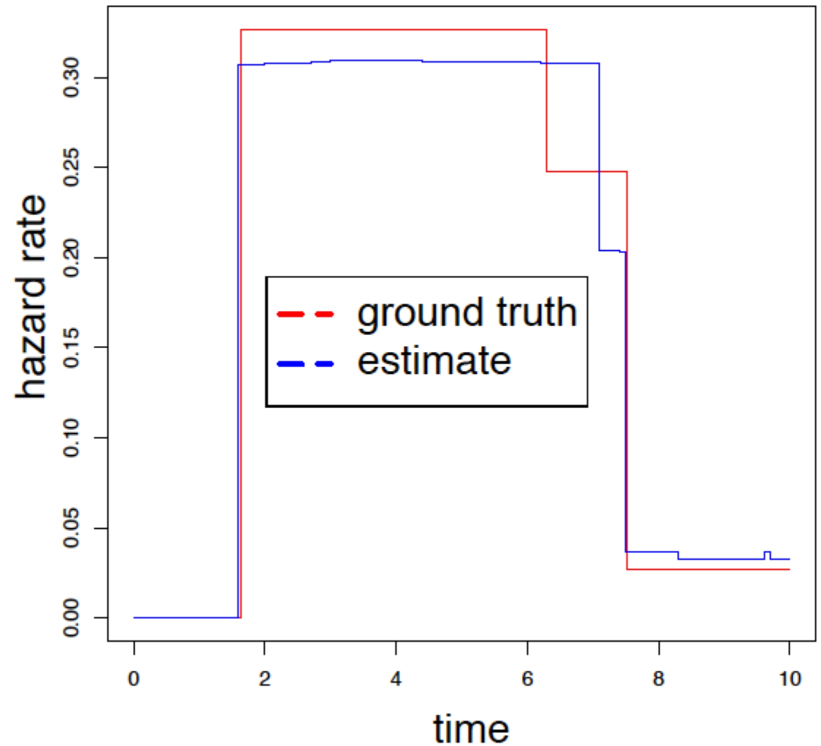
3. Our solution: survival analysis + trend filtering

4. **Results on simulation and real data**

Simulated example: recovery against the ground truth

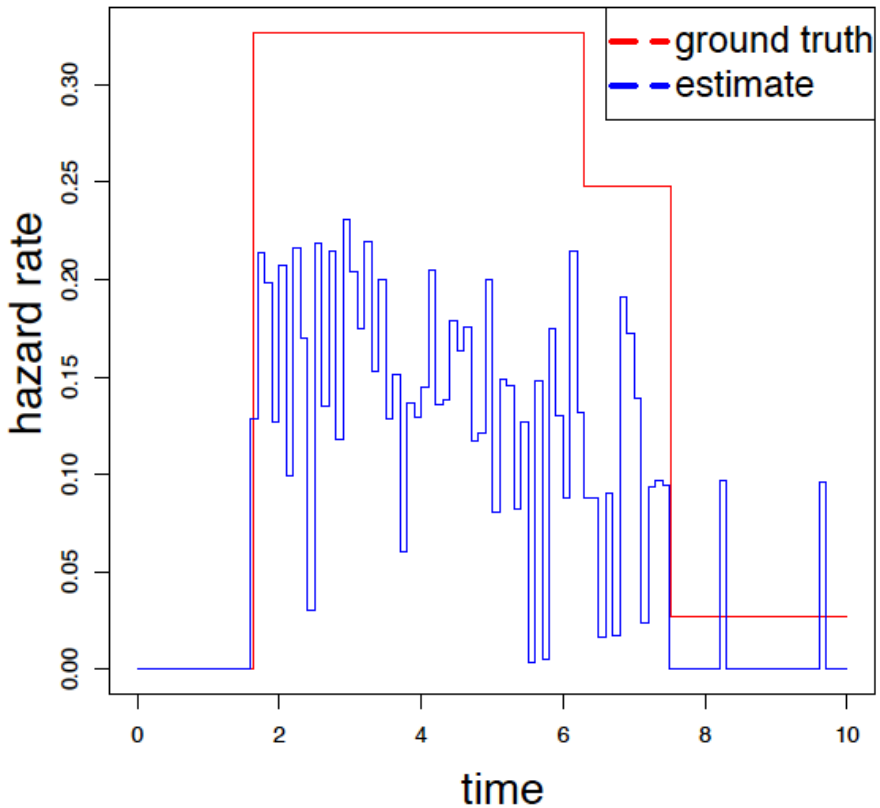


TV-penalty

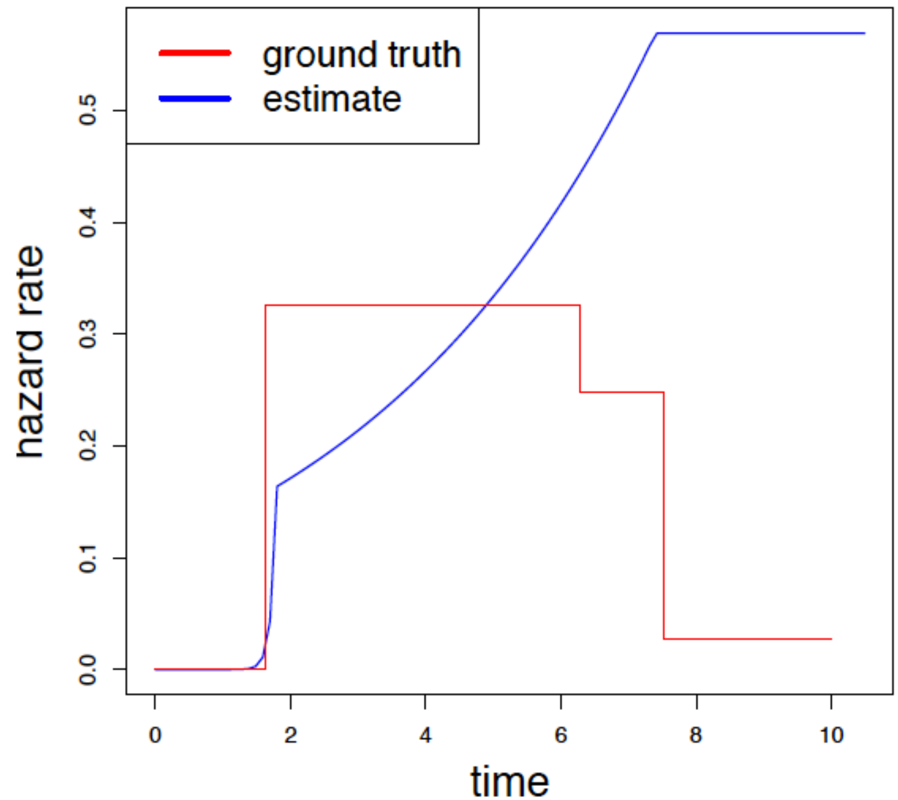


TV-log penalty

Simulated example: recovery against the ground truth



Unregularized

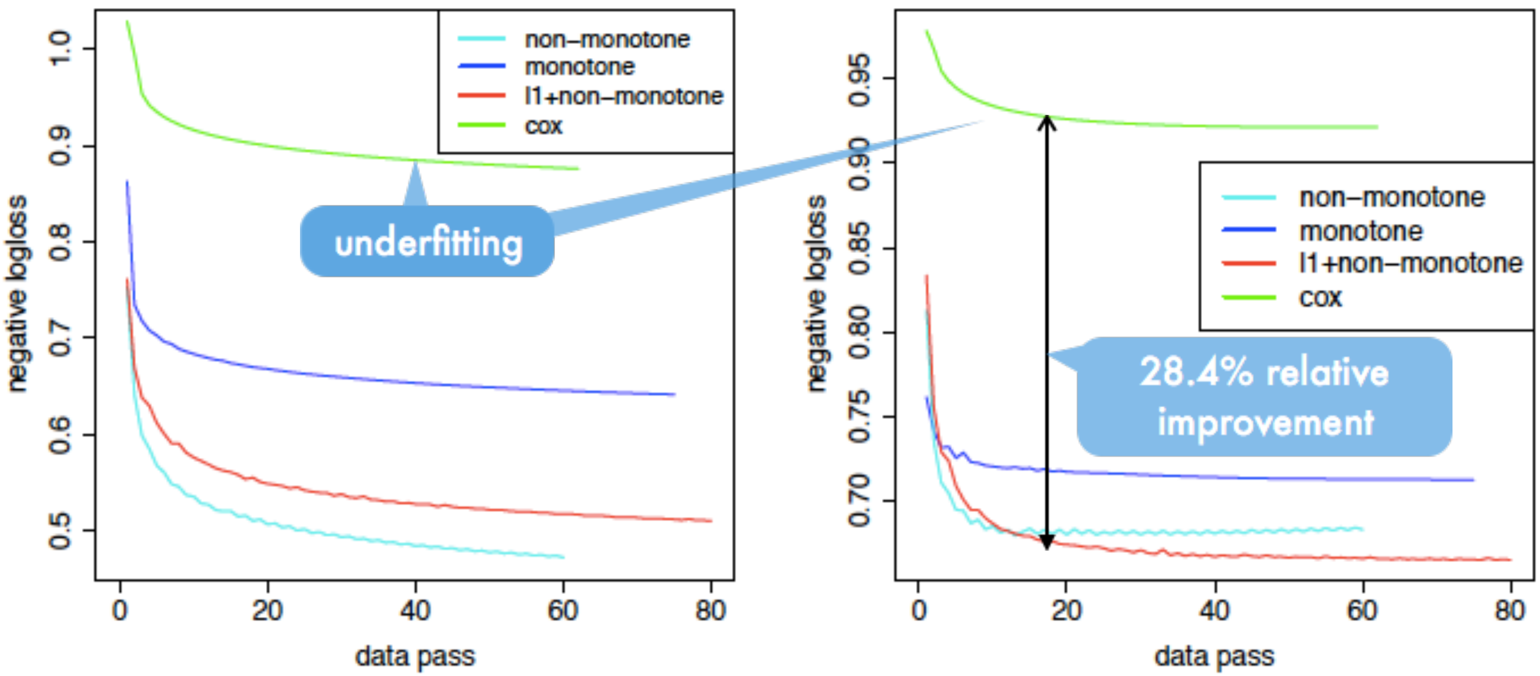


Polsplines in R

Experiments on millions of sites and millions of features, from 2010-2014.

	non-monotone	monotone	l1+nonmonotone	Cox
Parameter Size	$2 \cdot 10^6$	$4.04 \cdot 10^5$	$5.16 \cdot 10^5$	$1.59 \cdot 10^5$

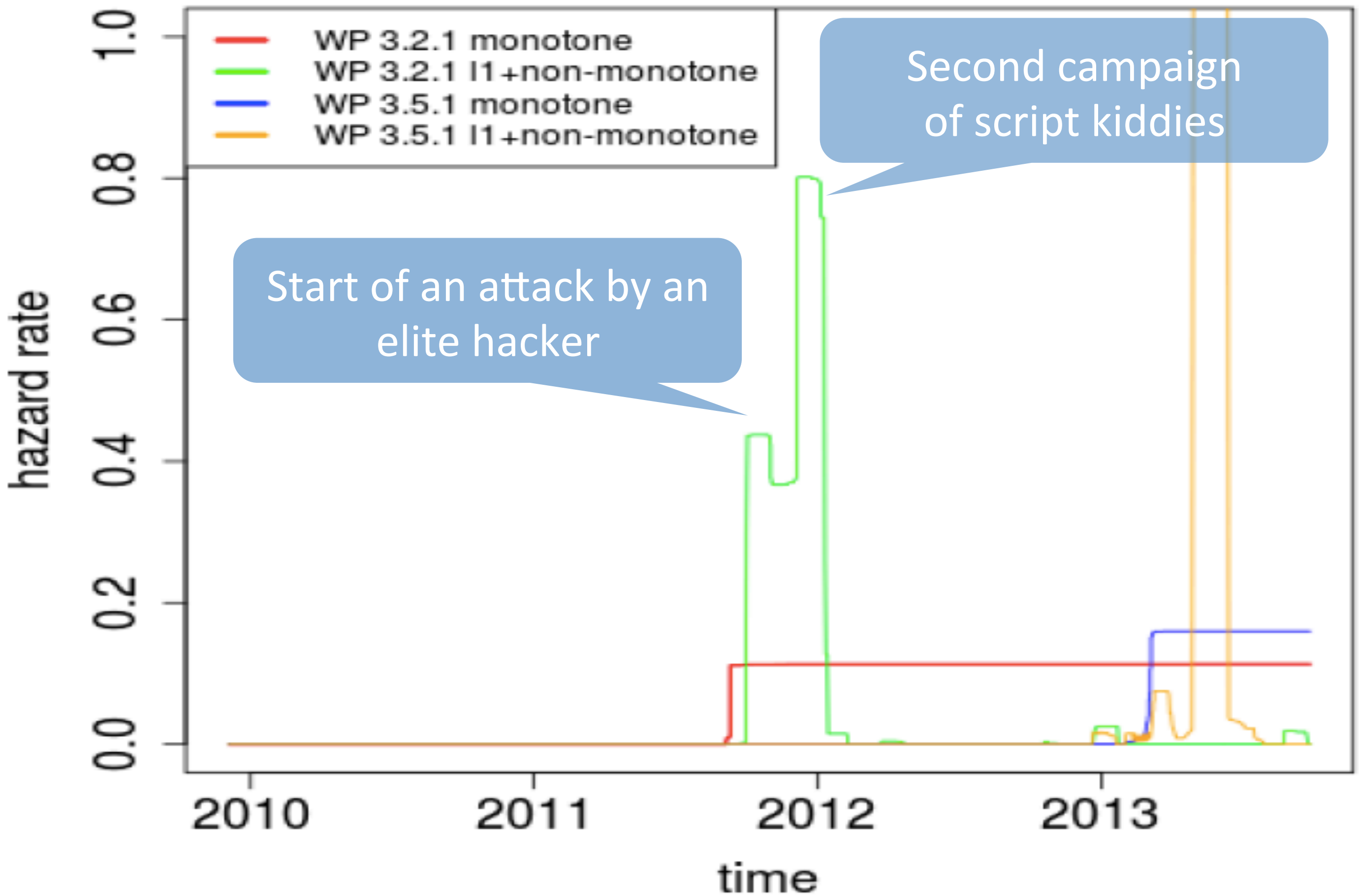
Table 1: Empirical model size estimated by different statistic models.



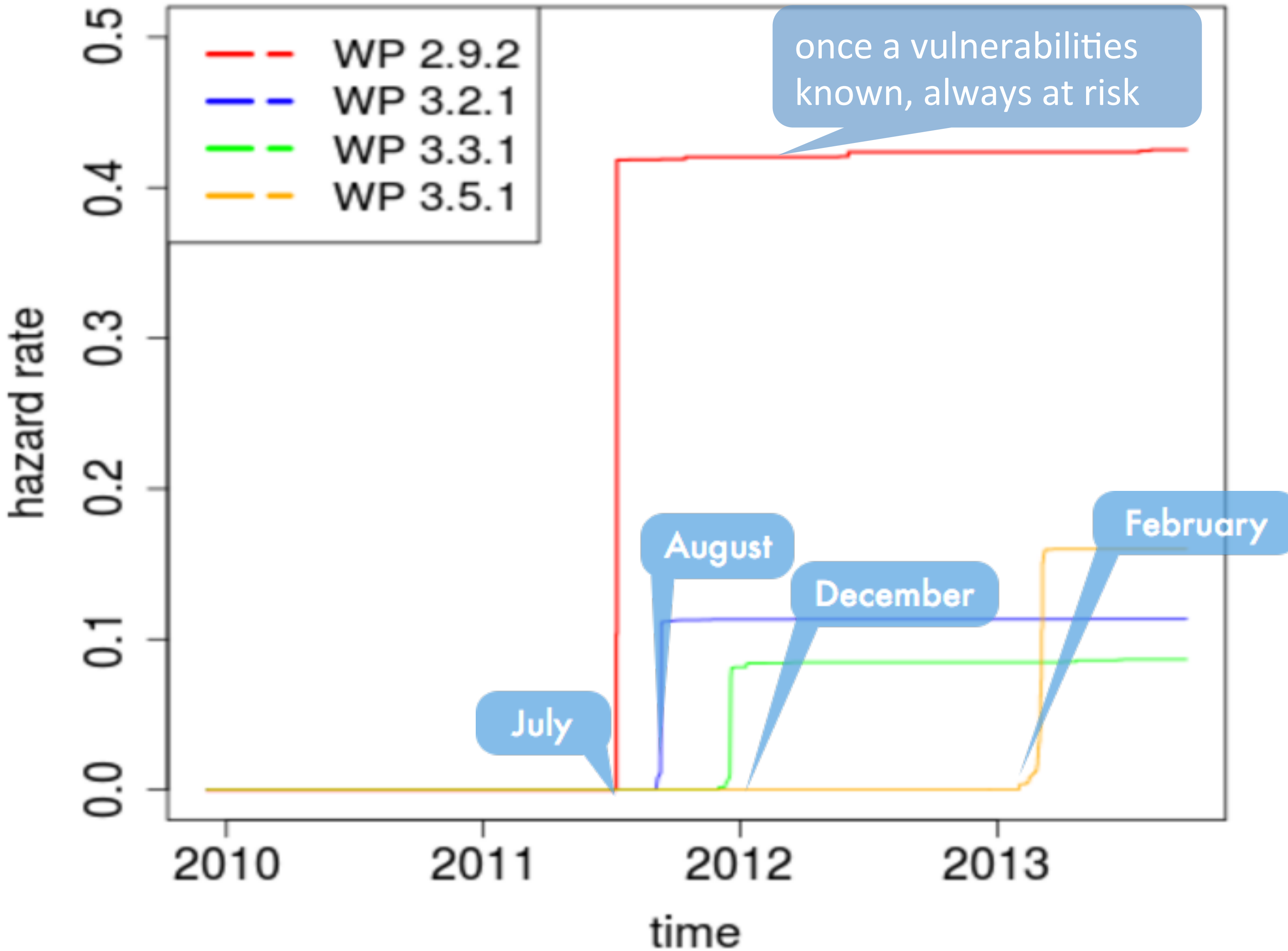
Training error **Test error**

Case study: Worldpress features

- Attackers tend to work in batches



Interpreting the monotone model



Other applications?

- User dropout rate estimation
  - Check responses of groups of people to certain promotions.

- Alipay.com data from Ant Financial.
  - Active user if log in for 7 days in a row.
  - Otherwise considered dropped out.
  - Data of 4 million users (1% of the Alipay users)

Results on the Alipay Data Set

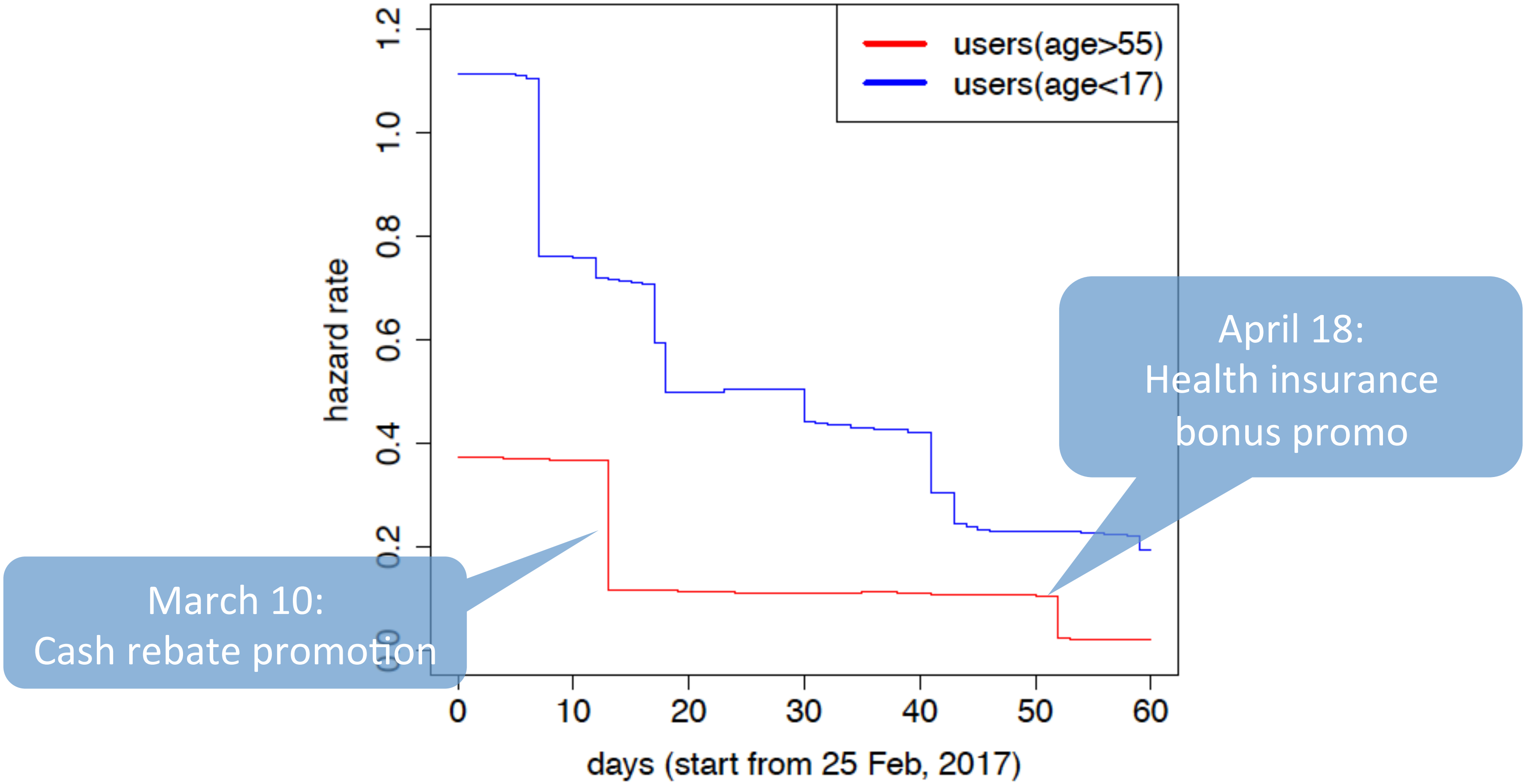


FIGURE 9. *Hazard rate on different features related to the ages of users.*

Conclusion

- Using 3x effective parameters, our model significantly outperforms the classic Cox model in prediction accuracy.
- Interpretability: Allows us to attribute hacks to features, and specific exploits.
- Scalability: faster and more locally adaptive than existing time-varying models.

Open problems

- Statistical properties:
  - Consistency and sample complexity of the model.
  - Implicit sparsity regularization? Sublinear dependence in d?

- Computational properties:
  - Nonconvex, but convergence to near global minima under statistical assumptions?

- Application:
  - Use higher order trend filtering on other survival analysis problems, e.g., marriage, divorce…

Thank you for your attention!

Ziqi

Alex

Kyle

Qinghua

Code/demo available at:
https://github.com/ziqilau/Experimental-HazardRegression