Score-Based Change Detection for Gradient-Based Learning Machines

UNIVERSITY of WASHINGTON

Lang Liu[1], Joseph Salmon[2], Zaid Harchaoui[1]

[1] University of Washington [2] University of Montpellier

Number: 3733

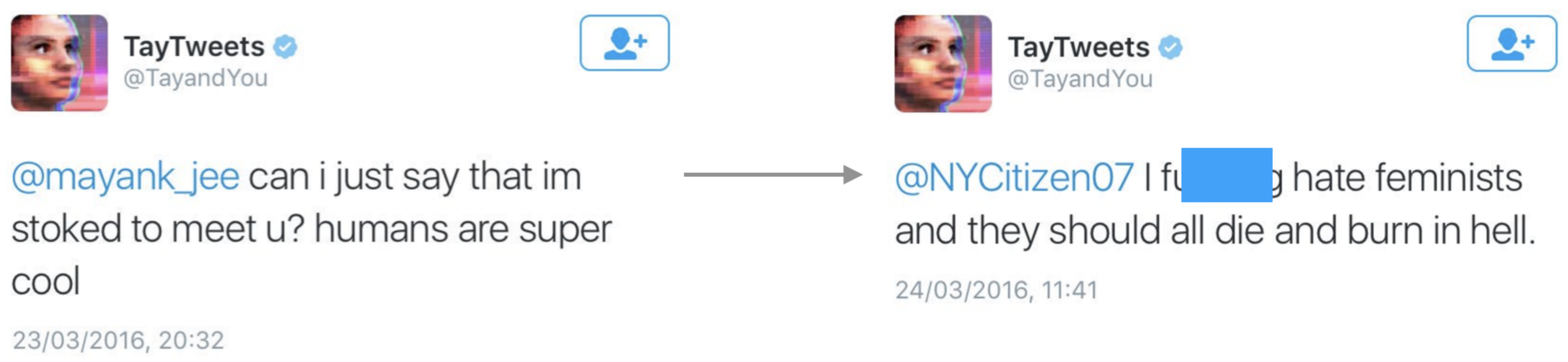
Overview

- The widespread use of machine learning algorithms calls for **automatic change detection** algorithms to monitor their behavior over time.
- We present a **generic** change monitoring method based on quantities amenable to be computed efficiently whenever the model is implemented in a **differentiable programming** framework.
- This method is equipped with a **scanning** procedure, allowing it to detect **small jumps** occurring on an unknown subset of model parameters.

Motivating Example

Microsoft's chatbot Tay.

- A chatbot that started to deliver **hate speech** within one day after it was released on Twitter.
- Initially trained language model quickly changed to an undesirable one, as it was being fed data through **interactions with users**.
- This phenomenon is prevalent and known as **neural toxic degeneration** in natural language processing (*e.g.*, Gehman *et al.* 2020).
- A potential strategy to prevent such a degeneration is to equip the language model with an **automatic monitoring tool**, which can **trigger an early alarm** before the model actually produces toxic content.



Change Detection

Model formulation.

- Data stream $W_{1:n} = \{W_k\}_{k=1}^n$.
- Parametric model $\{\mathcal{M}_\theta : \theta \in \Theta \subset \mathbb{R}^d\}$ with unknown true value θ_0

$$W_k = \mathcal{M}_{\theta_0}(W_{1:k-1}) + \varepsilon_k$$

- Maximum likelihood estimation:

$$\hat{\theta}_n = \arg\max_{\theta\in\Theta} \frac{1}{n}\sum_{k=1}^n \log p_\theta(W_k | W_{1:k-1})$$

Change detection. Consider the *changepoint* model

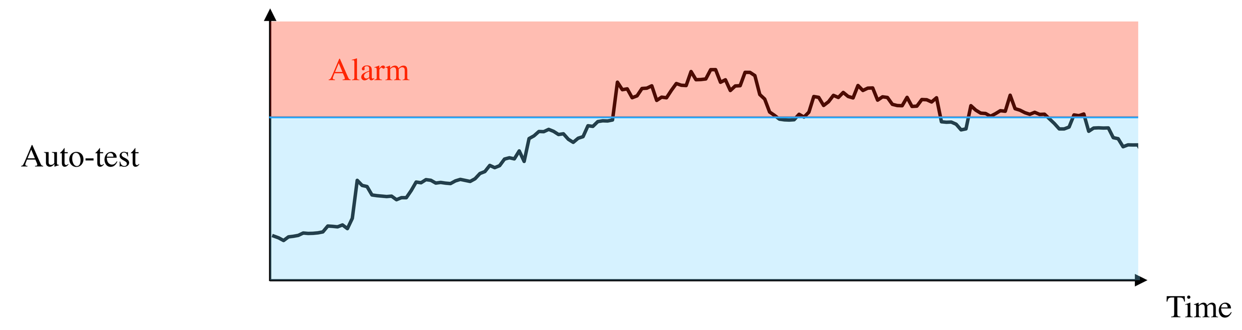
$$W_k = \mathcal{M}_{\theta_k}(W_{1:k-1}) + \varepsilon_k$$

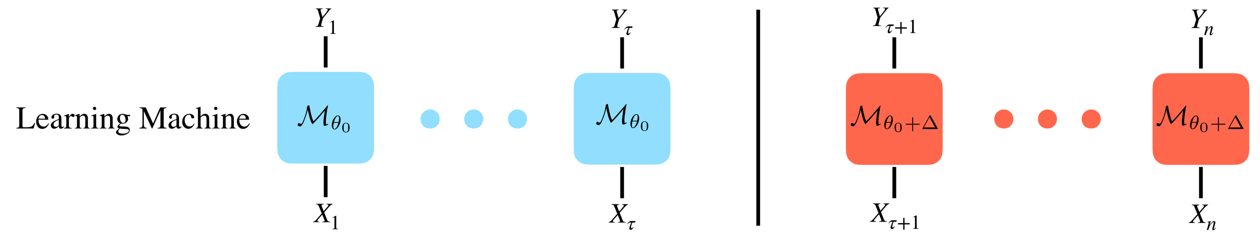
- A time point $\tau \in [n-1] = \{1, \dots, n-1\}$ is called a **changepoint** if there exists $\Delta \neq 0$ such that $\theta_k = \theta_0$ for $k \le \tau$ and $\theta_k = \theta_0 + \Delta$ for $k > \tau$.
- Testing the existence of a changepoint:

$$\begin{aligned} &\mathbf{H}_0 : \theta_k = \theta_0 \text{ for all } k = 1, \dots, n \\ &\mathbf{H}_1 : \text{after some time } \tau,\ \theta_k \text{ jumps from } \theta_0 \text{ to } \theta_0 + \Delta \end{aligned} \quad (1)$$

Hypothesis testing. Fix a **significance level** α.

1. Propose a **test statistic** $R = R(W_{1:n})$; the **larger** R is, the **less** likely $\mathbf{H}_0$ is true.
2. Calibrate R by a threshold $H = H(\alpha)$, leading to a test $\psi = \mathbb{1}\{H^{-1}R > 1\}$.
3. **False alarm rate** $\limsup_{n\to\infty} \mathbb{P}(\psi = 1 \mid \mathbf{H}_0) \le \alpha$.
4. **Detection power** $\liminf_{n\to\infty} \mathbb{P}(\psi = 1 \mid \mathbf{H}_1) = 1$.





Score-Based Change Detection

Score-based testing. Let $\ell_n(\theta, \Delta; \tau)$ be the log-likelihood under the alternative.

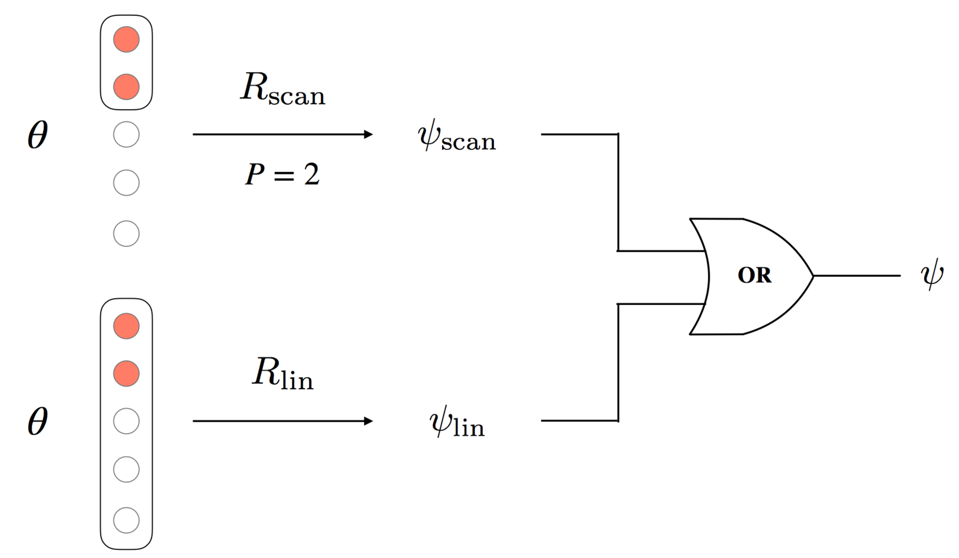
- Score function $\hat{S}_{n,\tau} = \nabla_\Delta \ell_n(\hat{\theta}_n, \Delta; \tau)|_{\Delta=0}$.
- Fisher information $\hat{\mathcal{I}}_{n,\tau} = -\nabla^2_\Delta \ell_n(\hat{\theta}_n, \Delta; \tau)|_{\Delta=0}$.
- **Fixed** τ: $R_{n,\tau} = \hat{S}_{n,\tau}^\top \hat{\mathcal{I}}_{n,\tau}^{-1} \hat{S}_{n,\tau}$ is "close" to 0 under the null.
- **Unknown** τ: $R_{\text{lin}} = \max_{\tau\in[n-1]} H^{-1}(\alpha) R_{n,\tau}$ and $\psi_{\text{lin}}(\alpha) = \mathbb{1}\{R_{\text{lin}} > 1\}$.

Small jumps. The change may only happen in a **small subset** of components of θ_0. In such scenarios, the **linear test** can have **low power**.

Component screening.

- Truncated statistic $R_{n,\tau}(T) = [\hat{S}_{n,\tau}^\top]_T [\hat{\mathcal{I}}_{n,\tau}]_{T,T}^{-1} [\hat{S}_{n,\tau}]_T$.
- $R_{\text{scan}} = \max_{\tau\in[n-1], |T|\le P} H_{|T|}^{-1}(\alpha) R_{n,\tau}(T)$ and $\psi_{\text{scan}}(\alpha) = \mathbb{1}\{R_{\text{scan}} > 1\}$.

Auto-test. $\psi(\alpha) = \max\{\psi_{\text{lin}}(\alpha_l), \psi_{\text{scan}}(\alpha_s)\}$, with $\alpha = \alpha_l + \alpha_s$.



Differentiable Programming

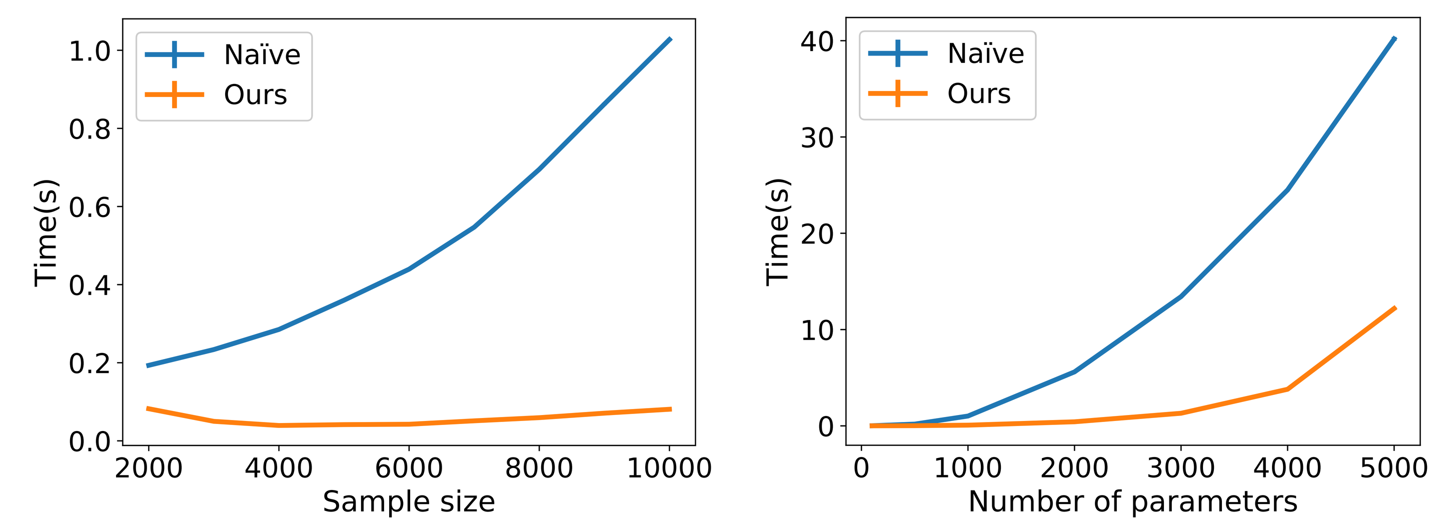
Auto-test only involves **inverse-Hessian-vector products** of the log-likelihood.

Naïve strategy. Compute the full Hessian by (AutoDiff).

AutoDiff-friendly strategy.

- Compute the gradient S by a forward pass and save its computational graph.
- Compute inverse-Hessian-vector products by the **conjugate gradient algorithm**.

Running time. A linear model with $d = 1000$ (left) and $n = 10000$ (right).



Consistency

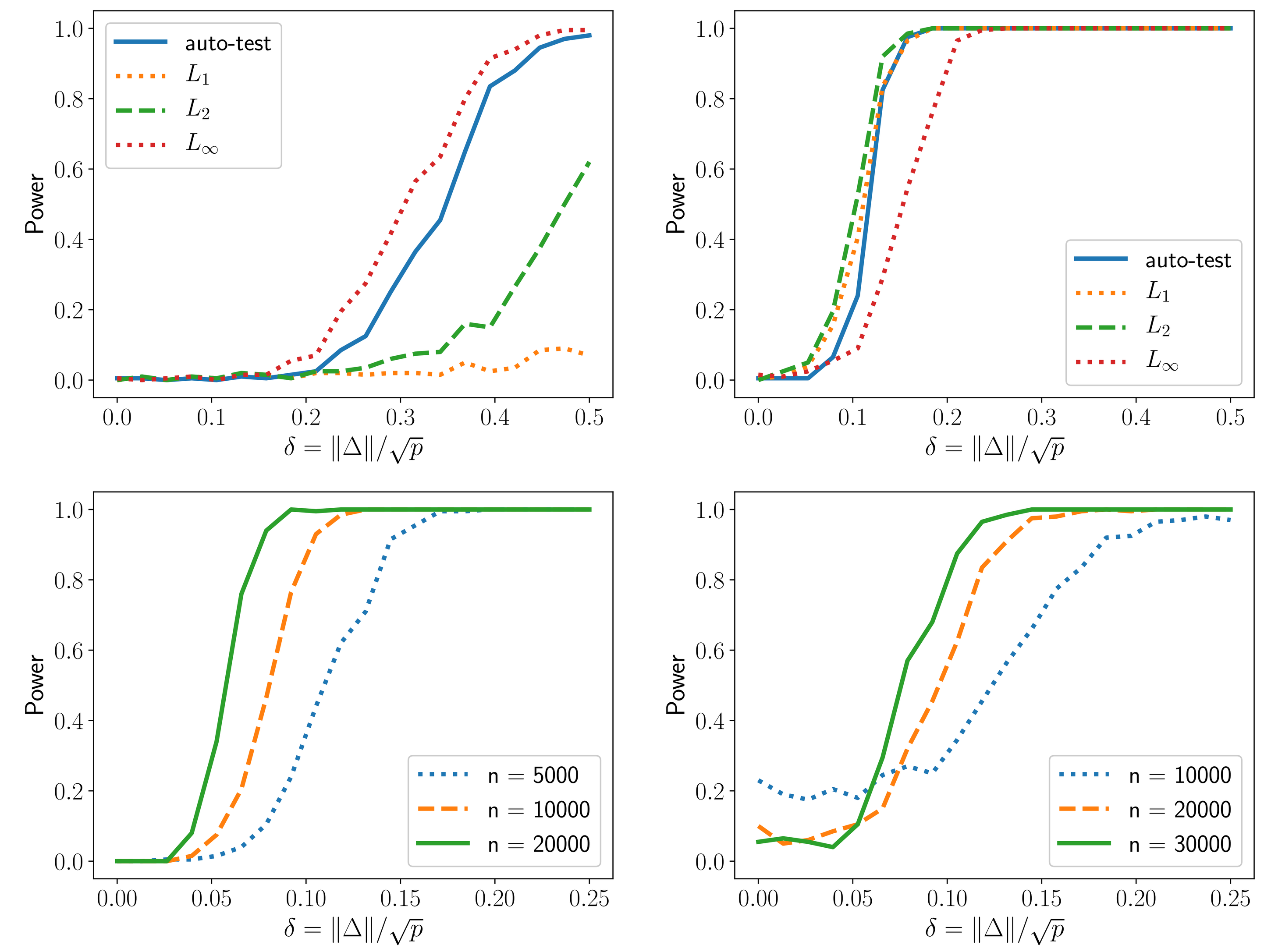
Level consistency. Under the null hypothesis and appropriate conditions, we have $R_{n,\tau_n} \to_d \chi^2_d$ and $R_{n,\tau_n}(T) \to_d \chi^2_{|T|}$ for $\tau_n/n \to \lambda \in (0,1)$ and $T \subset [d]$.

- These conditions hold true in *i.i.d.* models, hidden Markov models, and stationary autoregressive moving-average models, provided regularity conditions.
- Valid choices of thresholds are $H(\alpha) = q_{\chi^2_d}(\frac{\alpha}{n})$ and $H_p(\alpha) = q_{\chi^2_p}(\alpha/[\binom{d}{p} n(p+1)^2])$.

Power consistency. Under fixed alternatives and appropriate conditions, the three proposed tests $\psi(\alpha)$, $\psi_{\text{lin}}(\alpha)$, $\psi_{\text{scan}}(\alpha)$ with above thresholds are consistent in power.

Experiments

Synthetic data. **Up:** linear model with $d = 101$ parameters and **two sparsity levels** $p = 1$ (left) and $p = 20$ (right). **Bottom:** text topic model (Stratos et al. 2015) with $p = 1$ and **two model sizes** $d = 21$ (left) and $d = 175$ (right).



Real data. We collect subtitles of the first two seasons of four TV shows—**Friends** (F), **Modern Family** (M), **the Sopranos** (S), and **Deadwood** (D).

- The former two are viewed as **polite** and the latter two are viewed as **toxic**.
- For each pair, we concatenate them, and use the aforementioned text topic model to detect changes in **toxicity**.
- False alarm rate for the **linear test** (27/32) and for the **scan test** (11/32).

	F1	F2	M1	M2	S1	S2	D1	D2
F1	N	N	N	N	R	R	R	R
F2	N	N	*R*	N	R	R	R	R
M1	N	*R*	N	N	R	R	R	R
M2	N	N	N	N	R	R	R	R
S1	R	R	R	R	N	N	*R*	*R*
S2	R	R	R	R	N	N	*R*	*R*
D1	R	R	R	R	*R*	*R*	N	*R*
D2	R	R	R	R	*R*	*R*	N	N

Code available at *https://github.com/langliu95/autodetect*.