

Dynamic Evidence Disclosure: Delay the Good to Accelerate the Bad

Jan Knoepfle and Julia Salmi

Discussion

Yingni Guo
WIET at Yale

Recap of the Model: Big Picture

- Opening ideas evoke:
 - ▶ Kremer, Mansour, and Perry (2014)
 - ▶ Che and Hörner (2018)
- Common themes:
 - ▶ Information arises endogenously from early adopters
 - ▶ A designer seeks to maximize collective welfare
- Key distinctions:
 - ▶ Agents are forward-looking
 - ▶ The designer controls only when hard evidence is disclosed

Recap of the Model: Players and Payoffs

- A continuum of agents with heterogeneous discount rates
- Each agent chooses when to adopt a technology
 - ▶ The less patient adopt earlier
- Each agent's payoff depends on an unknown binary state $\omega \in \{G, B\}$:

$$v_B < 0 < v_G$$

- Common prior: $x_0 = \Pr[\omega = G]$

Recap of the Model: Learning and Design

- Let q_t denote the cumulative mass of adopters by time t
- Then \dot{q}_t is the flow of adoption at time t
 - ▶ If $\omega = G$, good news arrives at rate $\lambda_G \dot{q}_t$
 - ▶ If $\omega = B$, bad news arrives at rate $\lambda_B \dot{q}_t$
- The designer can withhold news and disclose it later
- Information is a public good, leading to under-provision: learning is too slow and too limited

Recap of Main Result

Theorem 1. *There exists an optimal policy z such that*

- *good evidence is disclosed at most once:* $z_t^G = \begin{cases} 0 & \text{if } t < \bar{t}, \\ q_{\bar{t}-} & \text{if } t \geq \bar{t}, \end{cases}$ with $\bar{t} \in (0, \infty]$,
 - *bad evidence is disclosed immediately:* $z_t^B = q_{t-}$ for all t .
- **Delaying news from t to $t + \Delta$:**
 - ▶ If bad news, adopters in $[t, t + \Delta]$ lose v_B : action margin
 - ▶ If good news, adopters at $t + \Delta$ lose $(1 - e^{-r\Delta})v_G$: timing margin
 - ▶ Delaying good news is more cost-effective
 - ▶ It also strengthens the signal of “no news” for adoption

Comments

- The paper is exceptionally clear, making my role as discussant a pleasure
- The authors' expertise in dynamics and exponential bandits is evident
- A clever and creative model that offers a tractable approach to a difficult problem
- The result that bad news is revealed immediately while good news is delayed resonates with many applications

Thoughts/Questions I

- Should learning depend on the flow of adoption or the stock of adopters?
- Buying a self-driving car?
News arrives based on the number of cars on the road (stock): $\lambda_{\omega} q_t$
- Watching a new movie?
News arrives based on new viewers at time t (flow): $\lambda_{\omega} \dot{q}_t$
- Getting a vaccine?
Possibly something in between
- Does this assumption matter?

Thoughts/Questions II

- Deepen the comparison with Kremer, Mansour, and Perry (2014) and Che and Hörner (2018)
- More than a shift from short- to long-run players
- Here, all agents prefer the optimal policy to transparent disclosure
- In contrast, early adopters lose under the optimal policy in KMP
 - ▶ Intuition: information is muddled to spur experimentation
 - ▶ Their strict IC constraints become binding
- Why does the optimal policy help everyone here? How general is this?

Thoughts/Questions III

- Deepen the comparison with Kremer, Mansour, and Perry (2014) and Che and Hörner (2018)
- *“Hiding good news is never optimal in Che and Hörner (2018), but is beneficial in our setting to deter agents from waiting.”*
- Yet CH also hide good news by pooling it with no news to recommend adoption
- In CH, good news leads to immediate adoption; here, it leads to delay
- Which assumptions drive this difference?

Thoughts/Questions IV

- Adding suspense when presenting results may help readers appreciate them
- The designer still has access to many instruments
- Some seem ex ante reasonable but are unused ex post – why?
- Theorem 1 focuses on sufficient instruments
- Are any instruments necessary?

Thoughts/Questions V

- Would the problem be trivial with identical discount rates?
- If not, what does heterogeneity add?
- Which results hold with homogeneous agents?
- Which rely on heterogeneity?

Thank you!